UNIVERSITÄT BREMEN

Zentrum für Technomathematik
AG Optimierung und Optimale Steuerung

MULTIDISCIPLINARY DESIGN OPTIMIZATION FOR SPACE APPLICATIONS

1. Gutachter: Prof. Dr. Christof BÜSKENS
2. Gutachter: Prof. Dr. Michèle LAVAGNA

Doktorarbeit von:
Annalisa RICCARDI

Akademisches Jahr 2011/2012

Abstract

Multidisciplinary Design Optimization (MDO) has been increasingly studied since the 80's in aerospace engineering with the main purpose of reducing monetary and schedule costs for the design of advanced aircrafts and space vehicles. The traditional sequential design approach of optimizing each discipline separately and manually iterating to achieve good solutions, is substituted by exploiting the interactions between the disciplines and concurrently optimizing every subsystem, in order to achieve global optimal designs.
The European Space Agency (ESA) proposed in 2009 to co-fund together with the Aerospace Engineering Department of Politecnico di Milano and the Center for Industrial Mathematics of Universität Bremen a three years research activity on MDO. The target of the joint PhD research was the development of a flexible software suite capable of concurrently optimizing the design of the subsystems of a rocket propellant launch vehicle employing novel MDO techniques.
The problem of efficiently integrating several disciplines in a single optimization problem leads to the design of the MDO architecture, the formulation of the overall optimization problem and the selection of a suitable optimization strategy.
The thesis is structured in five chapters. The first introductory chapter lays the foundations of the research activity in terms of motivations and historical background. A review of the current state of art of MDO architectures follows. These have been classified on the basis of the MDO problem formulation and decomposition level. The first is intended as the selection of the optimization variables set and statement of the interdisciplinary relations, whereas the second represents the break down structure of the multidisciplinary analysis. A suitable architecture has been designed for the MDO problem of Expendable Launch Vehicle (ELV), for which disciplinary models have been developed on the engineering side of the joint research activity. The possibility of having different trade-offs between vehicle cost, risk and performance, and the possibility of including integer and categorical design parameters have been considered for the selection of the optimization approach. Hence the optimization problem, in its most general formulation, is defined as a multi-objective Mixed Integer Non Linear Programming (MINLP) problem. Different techniques for global and local MINLP have been reviewed and presented in the third chapter. The possibility of combining the advantages of global and local searches have been exploited in both the MDO architecture and in the selected and self developed optimization methodologies. Four representative algorithms of the classes of single or multi-objective, deterministic or stochastic optimization techniques, have been identified as the most promising optimization approaches and compared on both analytic and MDO test cases. The stand alone ascent trajectory optimization problem is subject of the fourth chapter. The possibility of optimizing the trajectory separately from the vehicle design is considered as an alternative to the straightforward

approach of having just an optimizer on top of the multidisciplinary analysis. Considering this, techniques for fast payload assessment, based on semianalytic integration of the trajectory, are introduced.
The comparison among the optimization algorithms and the MDO architectures has been based on computational efficiency and performance criteria. Results have been critically analyzed in the fifth chapter to identify the most suitable optimization approach for the targeted MDO problem.

Acknowledgments

My greatest acknowledgments go to all the people that during my research helped me on the technical, financial and personal side.
In particular thanks to my supervisor Prof. Christof Büskens for always finding the time in his busy schedule for discussing about the progresses made and participating to all the project meetings around Europe that took place during these years. Thanks to Prof. Michèle Lavagna for reviewing my thesis and giving me the possibility of working in her research group in Politecnico of Milano for a very helpful exchange period of three months. Thanks to Francesco Castellini, we got to know each other by chance during an internship in ESA in 2008 and by that time we would never imagine to become PhD colleagues. Despites the different backgrounds we manage to strictly collaborate and compensate each other, in such a way that most of the time we were reading each other mind. Thanks to all the people from the former TEC-ECM section (now TEC-ECN): Guillermo Ortega, who supported me and Francesco through almost four years of collaboration, Antonio Rinalducci and Alvaro Martinez Barrio, who have been technical officers of the PRESTIGE activity and Sven Erb for its technical support and interest in the project. I would also like to thank Javier Ventura-Traveset and Carlos Lopez from the Education Office at ESAC in Madrid, who started the PRESTIGE program and sponsored the research activity on MDO.
There are a lot of people that provided us precious technical support and made the project to be successfully accomplished. Thanks to all the ESA expertises we have been in touch with during this "multidisciplinary" work. In particular thanks to Jens Kauffmann (ESA HQ, launchers directorate), Carlos Corral Van Damme (ESA CDF), Gianluigi Baldesi (ESA TEC-MSS), Marco De Rosa and Francesco Di Matteo (ESA TEC-MPC), Marco Chiappone (ESA TEC-QQD), Michel Van Pelt (ESA TEC-SYC), Cem Ozam and Patrick Rambaud from the Von Karman Institute of Fluid Dynamics, Massimiliano Bottacini (ESA HSF-ETS) and Fabio Caramelli (ESA HSF-PE). Special thanks go to Paolo Martino and Mitja Wöbbekind, MSc students from Politecnico di Milano and Universität Bremen who were responsible respectively for the development of cost and reliability models and for the branch and bound strategy for solving mixed integer non linear programming problems. Thanks to Chris Laurel from the precious help in software and graphic-related issues. Thank to the WORHP and NOMAD teams for the use of their optimization libraries, especially Tim Nikolayzik and Dennis Wassel for the continuous and efficient support on WORHP, as well as Prof. Kalyanmoy Deb and Abel Garcia Najera for providing the source code of the NSGA-II and MOACOr algorithms. Thanks to Dr. Matthias Knauer for reviewing my thesis and providing me enlightening comments, moreover for his helpfulness and scientific advices gave me during all these years. Thanks to Bodo Blume to be my office mate and supporting me in scientific and German-bureaucracy related issues. I cannot skip the administrative staff,

thanks to Tanja Rethemeyer for helping me in managing all the contracts and university papers. Thanks to all my current and past colleagues of the AG Optimierung und Optimale Steuerung, it was a pleasure to collaborate with all of you.

Now my special thanks go to my friends here in Bremen that gave me the most valuable moral support. In particular thanks to Sandra, Davide and Federico for being my backing in the good and in the bad moments. Thanks for always encouraged me and for being my "family" and my reference point here in Bremen.
Last but not least, thanks to my family and my life-time friends in Italy, for understanding my decision of continuing to study abroad and for being always present despite the distance.

A final thank to all the people I met in these years, you are too many to fit in such a small space, it was a joy to share my time with you, I wish our paths will cross again in the future.

Contents

Acronyms and Abbreviations 1

Chapter 1. Multidisciplinary Design Optimization 5
1.1. History and Background 5
1.2. What is MDO? 7
1.3. Why MDO? 10
1.4. MDO Framework and Commercial Tools 12
1.5. Application of MDO to Launch Vehicle Design 14

Chapter 2. MDO Techniques 17
2.1. MDO Problem Formulation 18
2.1.1. Multi-Disciplinary Feasible 20
2.1.2. Individual Disciplinary Feasible 21
2.1.3. All At Once 22
2.1.4. Comparison 23
2.2. MDO Decomposition methods 24
2.2.1. Hierachic Decomposition 25
2.2.2. Non Hierarchic Decomposition 26
2.3. MDO Architectures 27
2.3.1. Black-Box Optimization 27
2.3.2. Nested Optimization Loop 28
2.3.3. Multilevel Optimization by Linear Decomposition 30
2.3.4. Concurrent SubSpace Optimization 32
2.3.5. Bi-Level Integrated System Synthesis 36
2.3.6. Collaborative Optimization 38
2.3.7. Extension to Multi-Objective Problems 42
2.3.8. Bibliographic Comparison Considerations 43
2.4. Application of MDO Techniques to Launch Vehicle Design 44

Chapter 3. MDO Problem 53
3.1. Global Optimization 55
3.1.1. Deterministic Strategies 57
3.1.2. Stochastic Strategies 59
3.2. Multi-Objective Optimization 60
3.3. Local Optimization 65
3.3.1. Optimality Conditions 66
3.3.2. Optimization Algorithms 67

3.4. Application of Global and Local Optimization Strategies to MDO of Launch Vehicle 70
3.4.1. Global Single Objective Optimization Techniques 70
3.4.2. Global Multi-Objective Optimization Techniques 76
3.4.3. Local Optimization Techniques 85
3.4.4. Reverse Communication 86

Chapter 4. Optimal Control Problem 89
4.1. Optimal Control 89
4.2. The Ascent Trajectory Optimization Problem 91
4.2.1. Dynamic Model 94
4.2.2. Guidance Model 95
4.3. Multiple Shooting Techniques 97
4.4. Semianalytic Methods 98

Chapter 5. Numerical Results 109
5.1. Validation of Single Objective MINLP Techniques 110
5.1.1. Analytic Test Problems 110
5.1.2. MDO Test Problem 121
5.2. Validation of Multiple Objective MINLP Techniques 125
5.2.1. Analytic Test Problems 125
5.2.2. MDO Test Problems 153
5.3. Validation of the Trajectory Optimization Subproblem 166
5.3.1. Modeling 167
5.3.2. Test Case 171
5.3.3. Numerical Integration 171
5.3.4. Semianalytic Integration 177
5.4. Comparison of MDO Architectures 180
5.4.1. MDF vs IDF 180
5.4.2. BBO vs NOL 182
5.4.3. Local Refinements 184
5.4.4. Conclusions 185

Chapter 6. Conclusions and Future Developments 187

Appendix A. Commercial MDO Tools 191

Appendix B. SVAGO Software Tool 195

Bibliography 201

Acronyms and Abbreviations

Institutions and Projects:

AIAA	American Institute of Aeronautics and Astronautics
CASDE	Centre for Aerospace Systems Design and Engineering
ESA	European Space Agency
IIT	Indian Institute of Technology
ISSMO	International Society of Structural and Multidisciplinary Optimization
LaRC	Langley Research Center
MADORC	Multidisciplinary Aerospace Design Optimization Research Centre
MOB	Multidisciplinary Optimization Branch
NUDT	National University of Defense Technology
PRESTIGE	PRogram in Education for Space, Technology, Innovation and knowledGE
SVAGO	Space Vehicle Analysis and Global Optimization
TC	Technical Committee
WORHP	We Optimize Really Huge Problems

Mathematics:

AAO	All At Once
BB	Branch and Bound
BBO	Black Box Optimization
BBWORHP	Branch and Bound and WORHP algorithm
BFGS	Broyden-Fletcher-Goldfarb-Shanno
BLISS	Bi-Level Integrated System Synthesis
CC	Cumulative Constraint
CG	Conjugate Gradient
CO	Collaborative Optimization
COP	Coordination Optimization Problem
COSMOS	Collaborative Optimization Strategy for Multi-Objective Systems
CR	Consolidation Ratio
CSSO	Concurrent SubSpace Optimization

DGMOPSO	Double Grid Multi Objective Particle Swarm Optimization
DoF	Degree of Freedom
ECO	Enhanced Collaborative Optimization
GO	Global Optimization
GPS	Generalized Pattern Search
HD	Hierachic Decomposition
HGO	Hybrid Global Optimization
IDF	Individual Disciplinary Feasible
IP	Interior Point
KKT	Karush Kuhn Tucker
KS	Kreisselmeier-Steinhauser
LICQ	Linear Independence Constraint Qualification
LIP	Linear Integer Programming
LR	Local Refinement
MA	Memetic Algorithms
MADS	Mesh Adaptive Direct Search
MAO	Multidisciplinary Analysis and Optimization
MCO	Modified Collaborative Optimization
MDA	Multidisciplinary Design Analysis
MDF	Multi-Disciplinary Feasible
MDO	Multidisciplinary Design Optimization
MDR	Mutual Domination Ratio
Micro-GA	Micro Genetic Algorithm
MINLP	Mixed Integer Non Linear Programming
MOA	Multi-objective Optimization Algorithm
MOACO	Multi Objective Ant Colony Optimization
MOACOr	Multi Objective Ant Colony Optimization for continuous domains
MOGA	Multi-objective Genetic Algorithm
MOLD	Multilevel Optimization by Linear Decomposition
MOP	Multi Objective Problems
MOPCSSO	Multi-Objective Pareto Concurrent SubSpace Optimization
MOPSO	Multi-Objective Particle Swarm Optimization
MOSA	Multi-Objective Simulating Annealing
ND	No Decomposition
NHD	Non Hierarchic Decomposition
NLP	Non Linear Programming
NOL	Nested Optimization Loop
NOMAD	Nonsmooth Optimization by Mesh Adaptive Direct Search
NPGA	Niched Pareto Genetic Algorithm
NSGA	Non Dominated Sorting Genetic Algorithm
NSGA2	Non Dominated Sorting Genetic Algorithm V.2
OL	Optimization Layer
OSA	Optimum Sensitivity Analysis
PAES	Pareto Archived Evolution Strategy

PESA	Pareto Envelope-based Selection Algorithm
PI	Progress Indicator
PSDMADS	Parallel Space Decomposition of the Mesh Adaptive Direct Search
PSO	Particle Swarm Optimization
QP	Quadratic Programming
RSA	Response Surface Analysis
RSM	Response Surface Methodology
SA	Simulated Annealing
SLO	System Level Optimization
SPEA	Strength Pareto Evolutionary Algorithm
SQP	Sequential Quadratic Programming
SSA	System Sensitivity Analysis
SSD	System Sensitivity Derivatives
SSM	SubSystem Module
SSO	SubSpace Optimization

Engineering and Informatics:

ARV	Advanced Re-entry Vehicle
CAE	Computer Aided Engineering
CDO	Concurrent Design Optimization
CEA	Chemical Equilibrium with Applications
CRS	Costs, Risks and Scheduling
CSD	Central System Database
CSI	Central System Intelligence
DE	Design Environment
DSM	Design Structure Matrix
ECI	Earth Centered Inertial
ELV	Expendable Launch Vehicle
EoM	Equations of Motion
FD	Frozen Design
FDIR	Failure Detection, Isolation and Recovery
FGL	First Guess Layer
FLPP	Future Launchers Preparatory Programme
GPU	Graphich Programming Unit
GTO	Geostationary Transfer Orbit
GTOW	Gross Take-Off Weight
GUI	Graphical User Interface
IPD	Integrated Project Delivery
ISS	International Space Station
LCC	Life Cycle Cost
MPI	Message Passing Interface

NGL	Next Generation Launcher
ODE	Ordinary Differential Equation
OTS	Off The Shelf
PLF	Payload Fairing
REV	Re-Entry Vehicles
RLV	Reusable Launch Vehicles
STS	Space Transportation Systems
TO	Trajectory Optimization
TS	Trajectory Simulation
WER	Weight Estimation Relationships

CHAPTER 1

Multidisciplinary Design Optimization

Contents

1.1.	**History and Background**	**5**
1.2.	**What is MDO?**	**7**
1.3.	**Why MDO?**	**10**
1.4.	**MDO Framework and Commercial Tools**	**12**
1.5.	**Application of MDO to Launch Vehicle Design**	**14**

Multidisciplinary Design Optimization (MDO) is a new research branch of system engineering initiated in the late 1980s. The main necessity at the base of its development was to reduce monetary and schedule costs in the preliminary phase of a new design study. It is a new methodology that can be applied to the design of every engineering system that has two or more disciplines coupled together. It's been successfully employed in different design domains such as automotive, naval, construction and electronic [**1**], but its major success was registered for aerospace applications [**2**].
This chapter begins with a brief historical review of MDO focusing on the increase of needs and the evolution of theories and technologies in the fields of aerospace engineering, math and computer science, with the purpose of providing a clear overview of the environment where MDO originated, grew and developed to the current state of art. Afterwards, a collection of definitions about MDO given by the experts working in the field are quoted to give the reader a qualitative idea of the newly introduced methodology. Then, a contextualization of its application in the European aerospace industries and a comparison with the existing design methodologies is presented, to prove its effective usefulness in the real design world. A list of commercial existing MDO tools completes the overview of the subject. In the last section, the focus changes to the application of MDO techniques for the design of launch vehicles, selected as target vehicles for this study.

1.1. History and Background

What nowadays is called system engineering has its origins in the late 1930s. For many years the knowledge necessary to design an airplane resided in a single person who was not only the designer, but also an expert in all the other disciplines involved in the vehicle realization such as propulsion, aerodynamics, materials, structure and so on, being sometimes even the test pilot

himself. With the improvement of the techniques and development of new technologies, in the different fields of operation, there was a growth of specialized engineers in closed sectors. Hence the role of the chief designer shifted from being the expert in nearly all fields to ensuring the coordination of all inputs from disciplinary specialists. This is what is now called system engineer.
Due to the increasing interest in space and military programs in the late 1950s the leading role of the design engineer was undertaken by the analyst engineer. The analyst engineer is a specialist in the research of new ideas and technologies, pushing the limit of science achieving higher targets and performances. The part of the design engineer was than related to translate such innovative ideas into practice, losing his former leading role.
In the 1970s the initial query of better performance moved toward a balance between performance, life cycle costs and reliability. Moreover, the experience gained in the previous years led to the conviction that the system must not only be designed, but also optimized. This, together with the rapid growth of computational technologies as support of design engineers was at the base of the origin of MDO [**3**].
The scientific community started to talk about MDO, or equivalently Concurrent Design Optimization (CDO), as a scientific methodology as well as about its possible fields of applications in the 1980s. However MDO had conceptually been in use since the days of the Wright brothers. Jaroslaw Sobieski was the pioneer of decomposition methods for MDO applications: the complex system to be optimized is broken into smaller coupled subsystems concurrently executed in a predefined hierarchy [**4**]. This way he proved how it is possible to get closer to an optimal design solution with MDO approach where the common sequential engineering approach is likely to lead only to suboptimal designs.
In 1991, a Technical Committee (TC) for MDO was formed [**3, 5**] to gather the research community and provide an exchange of knowledge on methodologies and applications through the organization of conferences, publications and education programs. The MDO TC established in 1993 an MDO award, presented biennially at the Multidisciplinary Analysis and Optimization (MAO) Conference. This AIAA[1] Technical Award is " presented to an individual for outstanding contributions to the development and/or applications of techniques of Multidisciplinary Design Optimization in the context of aerospace engineering. In addition, persons who have developed or advocated tools, algorithms, methodologies, or processes, which, in and of themselves, are not explicitly multidisciplinary, but that are enablers of MDO are also candidates for the award" [**5**]. In the last years, the MDO TC has elected as recipients of the prize:

[1]American Institute of Aeronautics and Astronautics (AIAA)

Lucien A. Schmidt, Jr. (1994): for outstanding leadership and for pioneering the development, teaching and application of multidisciplinary optimization methods for engineering design

Jaroslaw Sobieski (1996): for pioneering fundamental research of multidisciplinary design optimization methods and for successfully advocating their acceptance in the Aerospace community,

Raphael T. Haftka (1998): for voluminous contributions to the Multidisciplinary Design Optimization theory and practice and for educating a cadre of MDO users,

Vipperla B. Venkayya (2000): for structural and aeroelastic optimization methods and tools,

Garret N. Vanderplaats (2002): for outstanding service to the aerospace community and for distinguished contributions to the vision, theory, and practice of multidisciplinary analysis and design optimization,

Prabhat Hajela (2004): for seminal contributions to the development and adaptation of soft computing methods for multidisciplinary design, and for leadership in promoting MDO education at the national and international levels,

Ramana Grandhi (2006): for outstanding leadership and enduring contributions to Multidisciplinary Design Optimization including methods adopted in commercial software for adaptive multipoint approximations, uncertainty quantification, and reliability based design of structures,

Ilan M. Kroo (2008): for founding and revolutionary technical contributions to the field of multidisciplinary optimization as well as pioneering and successful efforts in bringing MDO into practical industrial applications,

Achille Messac (2010): for pioneering research in multidisciplinary design optimization including control structure integrated design and physical programming, and for outstanding and visionary leadership in the aerospace community.

Most of the work undertaken by Sobieski and Kroo is at the base of the presented research work and will be discussed in detail in the dedicated chapters.

1.2. What is MDO?

As no proper definition for MDO exists, several quotes are presented here, which cover various aspects of requirements and benefits of this technique:

"A methodology for the design of complex engineering systems and subsystems that coherently exploits the synergism of mutually interacting phenomena." (NASA Langley Research Center (LaRC) Multidisciplinary Design Optimization Branch (MDOB)).

"Optimal design of complex engineering systems which requires analysis that accounts for interactions amongst the disciplines (or parts of the system) and

which seeks to synergistically exploit these interactions."
(AIAA MDO Technical Committee (TC)).

"How to decide what to change, and to what extent to change it, when everything influences everything else."
(AIAAMDO Technical Committee (TC)).

Multidisciplinary Design Optimization can be applied to the design of complex systems which present two ore more coupled disciplines. A system is said to be coupled, either when two ore more disciplines share the same design variables, or when the output of one discipline matches the input of another discipline. The disciplines with the greater impact on the overall design are identified. For each discipline, a set of representative design variables is determined and the interactions of the coupled variables between subsystems are exploited. In this way the different disciplines are considered concurrently, so that the effects of changes in a subset of the design variables are reflected on all the others and several design alternatives can be explored at the same time.

A system can be constituted of an arbitrary number of disciplinary subsystems, $N_{SS} \in \mathbb{N}$. $X \in \Omega \subseteq \mathbb{R}^{n_x}$ is the set of design variables in the design search space Ω, which is a subset of the real space with $\dim(\Omega)=n_x$. Denoting by $\mathcal{I} = \{1, ..., n_x\}$ the complete set of indexes, it is possible to define the projection for a subset of indexes $\mathcal{J} = \{j_1, ..., j_{n_j}\} \subseteq \mathcal{I}$

$$\mathcal{P}_{\mathcal{J}} : \mathbb{R}^{n_x} \to \mathbb{R}^{n_j},$$

that maps

$$(x_1, ..., x_{n_x}) \mapsto (x_{j_1}, ..., x_{j_{n_j}}).$$

Definition 1.2.1. For all sets of indexes $\mathcal{J} \subseteq \mathcal{I}$ let define the real subset

$$\Omega_{\mathcal{J}} = \{\mathcal{P}_{\mathcal{J}}(X) \,|\, X \in \Omega\} \subseteq \mathbb{R}^{n_j}.$$

Definition 1.2.2. For every set of indexes $\mathcal{J} \subseteq \mathcal{I}$ and $\mathcal{K} \subseteq \mathcal{I}$ it is possible to define an operation $\otimes$ between the two sets $\Omega_{\mathcal{J}}$ and $\Omega_{\mathcal{K}}$ as

$$\Omega_{\mathcal{J}} \otimes \Omega_{\mathcal{K}} = \{\mathcal{P}_{\mathcal{J}\cup\mathcal{K}}(X) \,|\, X \in \mathbb{R}^{n_x} \wedge \mathcal{P}_{\mathcal{J}}(X) \in \Omega_{\mathcal{J}} \wedge \mathcal{P}_{\mathcal{K}}(X) \in \Omega_{\mathcal{K}}\} \subseteq \mathbb{R}^{|\mathcal{J}\cup\mathcal{K}|}.$$

Analogously the same operation between two elements of the subsets, $X_j \in \Omega_{\mathcal{J}}$ and $X_k \in \Omega_{\mathcal{K}}$ is defined as

$$X_j \otimes X_k = \{\mathcal{P}_{\mathcal{J}\cup\mathcal{K}}(X) \,|\, X \in \mathbb{R}^{n_x} \wedge \mathcal{P}_{\mathcal{J}}(X) = X_j \wedge \mathcal{P}_{\mathcal{K}}(X) = X_k\}.$$

Now it is possible to make a distinction between local and shared variables. A local design variable is a design parameter that is involved in the analysis of a single discipline, while a shared one is in common between two or more subsystems. Hence, it is possible to rewrite the variables vector as

$$X = X_s \otimes X_l$$

where $X_s \in \Omega_{\mathcal{I}_s}$ are the shared design variables and $X_l \in \Omega_{\mathcal{I}_l}$ the local ones. $\Omega_{\mathcal{I}_s}$ and $\Omega_{\mathcal{I}_l}$ are respectively the projection of Ω on the subspaces $\mathbb{R}^{n_s}$ and

$\mathbb{R}^{n_l}$, where $n_s = |\mathcal{I}_s|$ and $n_l = |\mathcal{I}_l|$. To simplify the notation from now on the projection of Ω for a corresponding set of indexes $\mathcal{I}_i$ is indicated as

$$\Omega_i := \Omega_{\mathcal{I}_i}.$$

Similarly, narrowing down to a single discipline, X_i can be defined as the set of design variables involved in the analysis of the i-th discipline, $\mathcal{I}_{s_i}$ and $\mathcal{I}_{l_i}$ as the subset of indexes respectively of its shared and local variables, $X_{s_i} \in \Omega_{s_i}$ as the design parameters that the discipline shares with other subsystems and $X_{l_i} \in \Omega_{l_i}$ as the local ones. It then stands that:

$$X_i = X_{s_i} \otimes X_{l_i}$$

$$\begin{aligned} \mathcal{I}_{s_i} \cap \mathcal{I}_{s_j} \neq \emptyset, \qquad & \text{for some } i, j \in \{1, ..., N_{SS}\} \\ \mathcal{I}_{l_i} \cap \mathcal{I}_{l_j} = \emptyset, \mathcal{I}_{s_i} \cap \mathcal{I}_{l_i} = \emptyset \quad & \forall i, j = 1, ..., N_{SS}, \quad i \neq j. \end{aligned}$$

As stated above, there are also quantities computed through governing equations within the disciplinary analyses which are shared between subsystems. The set of such variables can be denoted as

$$Y = \otimes_{i,j=1}^{N_{SS}} y_{ij}$$

with $Y \in \mathbb{R}^{n_y}$ and where y_{ij} represent the output variables of the i-th discipline that are needed as input values of the j-th discipline. In the most general case the system is fully coupled, it means that for all the disciplines D_i, $i = 1, ..., N_{SS}$

$$X_i = X_{s_i}, \qquad y_{ij} \neq \emptyset, \quad \forall j = 1, ..., \in N_{SS}$$

the design variables of each subsystem are shared with the other disciplines and every subsystem sends its outputs $Y_i = \otimes_{j=1}^{N_{SS}} y_{ij}$ as inputs to all other subsystems.
If there is a two-way coupling among subsystems, i.e. $y_{ij} \neq \emptyset$ and $y_{ji} \neq \emptyset$ for some $i, j \in \{1, ..., N_{SS}\}$ the analysis of the design requires iterations between the two disciplines to reach an agreement. Besides, $P \subseteq \mathbb{R}^{n_p}$ is the set of constant parameters. The model is then linked to an optimization strategy which steers the design variables toward feasibility and optimality, where $f : \Omega \times \mathbb{R}^{n_y} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_{\text{obj}}}$ and $c : \Omega \times \mathbb{R}^{n_y} \times \mathbb{R}^{n_p} \to \mathbb{R}^m$ represent the objective and constraint functions. In Figure 1.1 an example of fully coupled system is given.

The MDO methodology is not intended as a "push a button" design process. The human intuition and experience play an important role. The process is a continuous question and answer iteration between user, model and optimizer. An a posteriori sensitivity analysis can identify the variables that play a prominent role in the optimization problem, while an a priori analysis determines the critical disciplines to accomplish the selected objectives (as for example the propulsion and aerodynamics for the achievement of fuel consumption performance). The user intervention is necessary not only in the pre and post process phase of the MDO process. The user is involved also in the optimization process itself, by analyzing the current optimal solution

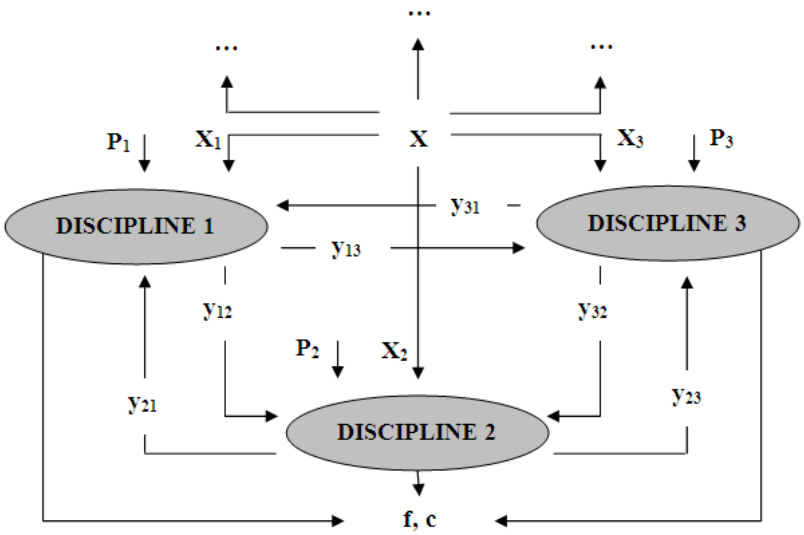


FIGURE 1.1. Schematic of fully coupled disciplines.

and operating on the variables bounds to steer the optimizer toward more promising regions of search.
MDO won't substitute the traditional design approach, but it can be a valid support for engineers in the preliminary design phases, with the goal of knocking down time and costs as well as achieving more competitive design solutions through the optimization approach.

1.3. Why MDO?

The motivations that made MDO growing as a technique are of economical and qualitative nature. The competitive market of the late 1980s, pushed toward the improvement of the productivity and the quality of the offered products.
The sequential design approach, which provides design solutions through the manual iteration between disciplinary simulations, has been proved to lead to sub-optimal designs [**4**]. In a sequential design, the value of a design variable which is shared between different subsystems can be frozen during the analysis of one discipline, not allowing the others to vary it and preventing the exploration of the entire search space, likely losing global optimal solutions. MDO doesn't present this limitation and, by concurrently acting on all modeled variables, it can reach global optimal solutions.
It is necessary to go through the design process to understand how MDO can help improving not only the quality of the design solutions but also the industrial productivity. The development of new complex systems has different design stages: first the *conceptual* and *preliminary* phases, where the baseline of the final system is delineated, and then *detailed* design *manufacturing* and *testing* phases where the configuration is frozen and the production of the final product begins [**3**].

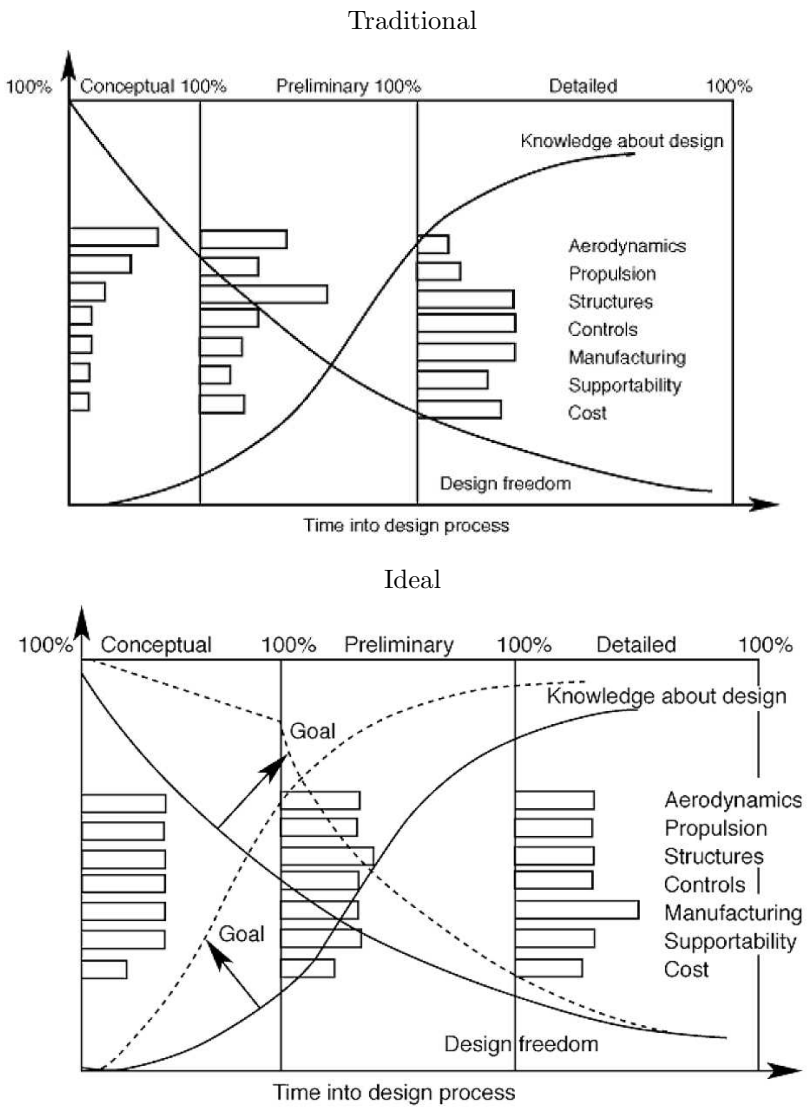


FIGURE 1.2. Approaches to product development: design freedom vs knowledge about the product in aerospace design [**3**].

Multidisciplinary Optimization can be efficiently used in the first two steps where the design freedom is higher, the final configuration is not yet frozen and the designer has more freedom to change it according to the objectives. Indeed, when the knowledge about the product design matures, the freedom of acting on the design itself decays, and the use of MDO techniques would be pointless for a design that is too constrained.
Moreover, the use of MDO in the initial phases can improve the distribution of the disciplinary analyses effort, ensuring a higher design freedom and greater knowledge about the system, as shown in Figure 1.2. Besides, the time spent in the detailed design phase can be shortened counting upon a stronger baseline as starting configuration.

The previous considerations are reflected also in the estimation of the final costs. It has in fact been shown that most of the Life Cycle Cost (LCC) is determined early in the conceptual phase when, as stated before, the design freedom is higher, whereas it is difficult to significantly affect it with decision performed later [**3**].
In conclusion, it is possible to identify two main advantages in the use of MDO techniques for the design of complex engineering systems:

- MDO brings enhancements in the iterative design process, by drastically reducing the human effort and the related costs, especially considering the need of several disciplinary experts in a manual iteration process.
- MDO allows to perform conceptual design processes more efficiently, in terms of achievements of the design objectives and costs. The chances of developing competitive design proposals increase.

Even when compared with the most recent concurrent design laboratories, the MDO approach is expected to register significant improvements in time, costs and outcome of any conceptual study. The technical and economic considerations allow to justify the worldwide interest of the industries in MDO and constitute the main motivations of the presented research.
The applicability of MDO in preliminary studies is not the only opportunity for the new methodology. MDO can in fact be efficiently employed in industries also for the identification of components in existing systems that need to be enhanced for achieving better performances or lower costs, and to perform their modification.

1.4. MDO Framework and Commercial Tools

When MDO started to grow as a new methodology, the industrial need of having a robust and flexible software tool arose, and the experts in the field identified the key issues of the new strategy.
A first taxonomy for MDO was given by Sobieski in [**7**]. After the formation of the Technical Committee and the publication of two White Papers (1991-1998), the original subdivision in categories drawn by Sobieski was extended for the current industrial needs, perspectives and priorities. The elements composing any MDO process are generally divided into four groups [**2**]. For each of these, a short description is given below, together with the industrial needs originating from their implementation in a commercial tool:

Design Model Formulation and Solution. It is formed by the definition of the optimization problem objectives and constraints, differentiation of the optimization levels (decomposition methods), and selection of the best optimization algorithm. The main obstacles in the development of a generic MDO tool is to allow custom definition of objectives and constraints, as well as robust, global, efficient optimization strategies and flexible decomposition structures. These

should for example allow high fidelity models to be called periodically instead of at each iteration.

Analysis Capabilities and Approximations. This includes pre and post processing, integration of engineering models with different levels of accuracy, approximation methods for the correction of disciplinary models which cannot be directly called by the optimizer for robustness issues, parametric models to facilitate computation of model changes, and disciplinary sensitivity analyses. The main issue for the modeling part is to find a compromise between model simplicity and accuracy. Research on this side is currently focused on the investigation of more sophisticated approximation techniques.

Information Management and Processing. This embodies the software architecture, data managing, computation efficiency and visualization tools. The main issues are related to designing a software architecture which is capable of interfacing with existing commercial tools, to the massive parallelization of the processes and the definition of data standards. Besides, multidimensional results visualization methods are presently under investigation.

Management and Cultural Implementation. It is the reorganization of the company structure for using MDO in the early steps of a conceptual design. Critical aspects are the interaction between tools and the members of the Integrated Project Delivery (IPD) team, the validation of the results and the evaluation of benefits, training and teaching.

Only the first and the third category were investigated in the present research work, as outlined in the schematic representation of the taxonomy in Figure 1.3, for the development of a stand alone software tool.

A considerable variety of commercial and academic MDO tools is currently available on the market. In Annex A, a comprehensive list of MDO development environments is reported with a short description of their approach to the multidisciplinary analysis. Several aspects are covered in the review, such as flexibility to the integration of external discipline modules, optimization and metamodeling techniques, visualization and user interaction features, in order to contextualise the development of the new tool and detect the main innovative points.

The strength of any MDO tool relies in its flexibility. Flexibility is intended either as the possibility of including external disciplinary codes or the allowance of multiple MDO architectures. This means including multilevel optimization processes, redefinitions of the optimization problem in term of objectives, variables and constraints and lastly the adaptivity, from an optimization point of view, to solve a wide variety of optimization problems with continuous and discrete optimization variables, single or multi-objective, unconstrained or constrained. Moreover, the management of the input, output and internal data flows are further issues to be considered.

While most of the tools presented in the annex offer good flexibility in the

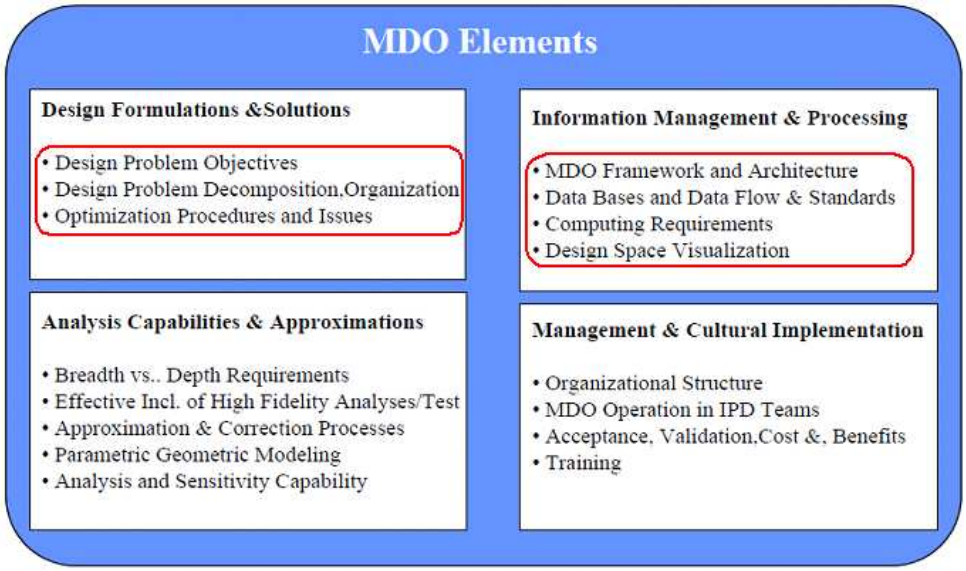


FIGURE 1.3. MDO Taxonomy [**2**].

integration of third party codes or external software, almost no effort has been spent on the optimization layer. None of these software allows the definition of nested optimization processes, executed in parallel with the top level optimizer. The optimization strategies included are mostly the best known algorithms for deterministic and stochastic optimization. Hybrid optimization approaches between the already present strategies are not envisaged. These are the software design details investigated during the realization of the present research work, from the side of mathematics and computer science. The purpose is not the commercialization of a competitive tool but more to analyze and compare different approaches to MDO not feasible with the available software, as well as to develop an ad hoc design environment for the specific design problem.

1.5. Application of MDO to Launch Vehicle Design

As stressed in Section 1.3, aerospace industries are the best recipients of the new MDO design methodology. Aerospace vehicles are complex and highly coupled disciplinary engineering systems. While MDO has already been successfully applied to aircraft design [**2**], its growth in the space sector is still slow.

When looking at the future of space exploration, the area with the highest potential for the development of new vehicles is surely space transportation and space launch systems, both for manned and unmanned scenarios. In Europe, the Future Launchers Preparatory Programme (FLPP) [**9**], kicked-off in 2004, is aimed at the development of technologies for the Next Generation Launcher (NGL) and the Advanced Re-entry Vehicle (ARV) project [**10**], for cargo and crew transportation to and from the International Space

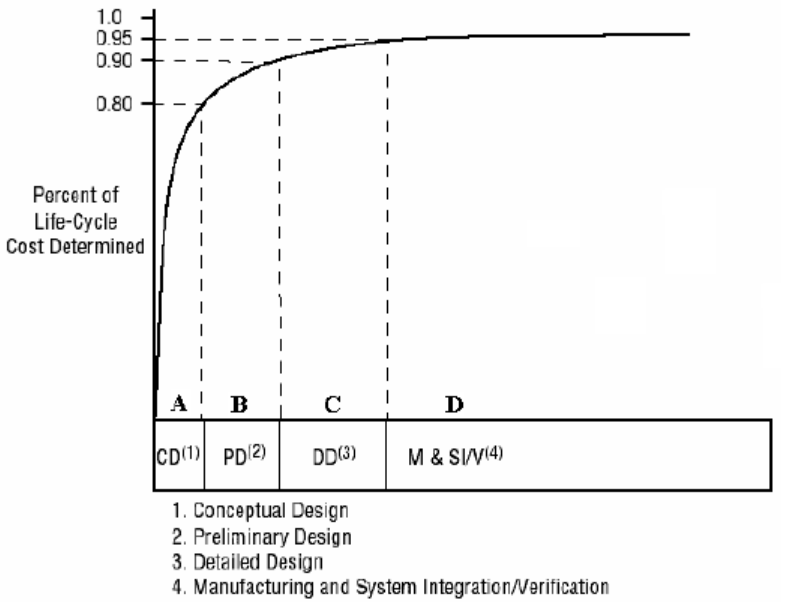


FIGURE 1.4. Percentage of System LCC as a function of the design phase, taken from [**6**].

Station (ISS).
The application of MDO techniques in the early design phase allows to drastically reduce the time required for conceptual studies and to increase the number of analyzed design alternatives, having the potential to identify new and more cost-effective concepts that would not be envisioned with the conventional approach. As announced in the previous section, it has been remarked in [**6**], as shown in Figure 1.4, that 80% of the total cost of a launch system is determined early in its design phase, exactly where MDO best applies.
In Figure 1.5, a general schema of the different launcher configurations, that could be of interest to space agencies for the design of advanced space transportations systems, is reported. Peculiar are the trade offs of the vehicle disciplines: the geometry shape can span between the traditional, lifting body or winged body, the vehicles can be expendable or reusable, it can carry manned or unmanned payload and have rocket, air breathing, liquid rocket, solid rocket or combined propulsion systems.
The European Space Agency (ESA) proposed in 2009 to co-fund together with the Aerospace Engineering Department of Politecnico di Milano and the Center for Industrial Mathematics of Universität Bremen a joint research in the field of Multidisciplinary Design Optimization under the PRogram in Education for Space, Technology, Innovation and knowledGE (PRESTIGE) grant.
The objective of the research activity was originally the development of a generic MDO Design Environment to deal separately with the design of rocket

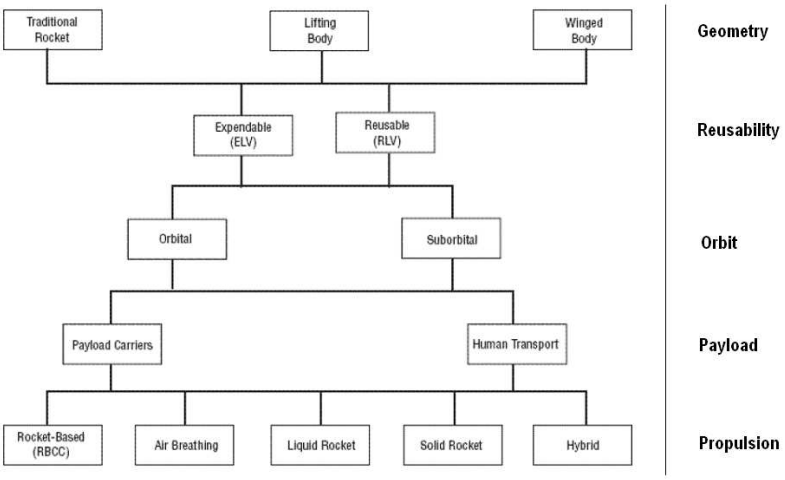


FIGURE 1.5. Launch Vehicle configuration tree [**8**].

propelled ELV and ballistic or glided Re-Entry Vehicles (REV), and in a second step with rocket propelled Reusable Launch Vehicles (RLV) capable of both ascending and re-entering the atmosphere in a suborbital or orbital flight. The developed environment has been named Space Vehicle Analysis and Global Optimization (SVAGO)[2]. While the framework has been developed in a generic way, only the disciplinary models related to ELVs for both conceptual and early preliminary levels of detail have been included and tested, leaving the MDO architecture as flexible and modular as possible for foreseen future extension of the tool to other space vehicle applications of European interest. A short description of the developed software, features and capabilities is given in Annex B.

[2]Italian of amusement or activity that distracts from the daily routine.

CHAPTER 2

MDO Techniques

Contents

2.1. MDO Problem Formulation **18**
2.1.1. Multi-Disciplinary Feasible 20
2.1.2. Individual Disciplinary Feasible 21
2.1.3. All At Once 22
2.1.4. Comparison 23
2.2. MDO Decomposition methods **24**
2.2.1. Hierachic Decomposition 25
2.2.2. Non Hierarchic Decomposition 26
2.3. MDO Architectures **27**
2.3.1. Black-Box Optimization 27
2.3.2. Nested Optimization Loop 28
2.3.3. Multilevel Optimization by Linear Decomposition 30
2.3.4. Concurrent SubSpace Optimization 32
2.3.5. Bi-Level Integrated System Synthesis 36
2.3.6. Collaborative Optimization 38
2.3.7. Extension to Multi-Objective Problems 42
2.3.8. Bibliographic Comparison Considerations 43
2.4. Application of MDO Techniques to Launch Vehicle Design **44**

Since the first steps toward multi-disciplinary design, a large number of techniques have been proposed to deal with the task of efficiently integrating several disciplines into a single design optimization environment. This chapter presents an overview of such approaches, reflecting the work during the late 1980s and 1990s of many researchers (Sobieski, Alexandrov, Haftka, Lewis, Braun, Kroo and Olds among the others) mainly at NASA's Multidisciplinary Optimization Branch (MOB) of Langley Research Center (LaRC) and its collaborating universities (Stanford University, Georgia Tech, Virginia Polytechnic Institute, North Carolina State University and University of Florida above all). The most investigated applicative area for these early studies was aeronautical design. In more recent years, research in the field has been extended to several other academic and industrial centers around the world, focusing more on practical applications (expanding from aeronautics to space and many other engineering areas) rather than on the development of brand new methodologies. Recent examples of these emerging

research centers in MDO are the Multidisciplinary Aerospace Design Optimization Research Centre (MADORC) of the Chinese National University of Defense Technology (NUDT) and the Centre for Aerospace Systems Design and Engineering (CASDE) of the Indian Institute of Technology (IIT), whereas, as already mentioned in Chapter 1, the most relevant coordinating international societies are the AIAA MDO TC and the International Society of Structural and Multidisciplinary Optimization (ISSMO). When starting the implementation of a MDO environment, the following are the most important aspects to take into account:

Effectiveness, i.e. probability of success of the optimization process, related to the capability of investigating all major system configurations without loosing possibly good solutions. Note that this is sometimes referred to as robustness in literature.

Efficiency, in terms of computational resources required to achieve convergence.

Reusability of existing disciplinary analysis and optimization codes.

Maintainability, intended as the easiness to add a new discipline or substitute an analysis code with another, for example to introduce a higher fidelity level.

Potential for parallel analysis, intended as the possibility to distribute the computational burden of the different disciplines among several processors.

This chapter is focused on the concepts of *problem formulation* and *problem decomposition*, intended as the mathematical formulation of the general MDO optimization problem. Particular attention is paid to the choice of the optimization variables set and the efficient arrangement of the modules inside the multidisciplinary analysis to exploit the interdisciplinary relation and perform fast model evaluation. All the MDO techniques available in literature are shortly presented and classified according to these two different aspects. A comparison of the different techniques based on the above criteria is accomplished, with the purpose of identifying the best MDO architecture to be applied to the specific case of multidisciplinary design of ELV.

2.1. MDO Problem Formulation

An interesting classification related to the MDO problem formulation is proposed in [**11**].

In order to fully understand the different formulations, the notation of Section 1.2 is retrieved and extended as in [**11**]. It is necessary to point out that in a generalization of the multidisciplinary design process, it is possible that a disciplinary input coming from the analysis of discipline D_j is mapped to the input required by discipline D_i, through interpolation or approximation techniques. If the output exactly matches the input value, the identity transformation is considered.

Focusing on one single discipline, as shown in Figure 2.1, the following elements are introduced as generalization of a MDO problem, where n_{X_i} is the number of input design variables of the discipline D_i, n_{M_i} is the number of inputs needed by D_i from the other subsystems and n_{U_i} the number of its outputs:

- $M_i \in \mathbb{R}^{n_{M_i}}$, inputs to the analysis of discipline D_i, needed from the other disciplines, M_{ji} represents the input needed by the i-th discipline from the j-th discipline. For simplicity of the notation and without loss of generality let consider $M_{ji} \in \mathbb{R}$ for all $i, j = 1, ..., N_{SS}$.
- $U_i \in \mathbb{R}^{n_{U_i}}$, outputs of the discipline D_i computed through the complete discipline analysis.
- $A_i : \mathbb{R}^{n_{X_i}} \times \mathbb{R}^{n_{M_i}} \to \mathbb{R}^{n_{U_i}}$, analysis of discipline D_i
$$U_i = A_i(X_i, M_i).$$
- $W_i : \mathbb{R}^{n_{X_i}} \times \mathbb{R}^{n_{M_i}} \times \mathbb{R}^{n_{U_i}} \to \mathbb{R}$: residual function of the equations solved in the analysis of the discipline D_i. These equations take the general form
$$W_i(X_i, M_i, U_i) = 0.$$
- $E_{ji} : \mathbb{R}^{n_{X_i}} \times \mathbb{R} \to \mathbb{R}$, $F_{ij} : \mathbb{R}^{n_{X_i}} \times \mathbb{R}^{n_{U_i}} \to \mathbb{R}$: respectively mapping of the input required for the evaluation of the disciplinary model A_i and mapping of its output needed by the evaluation of the other discipline analyses
$$M_{ji} = E_{ji}(X_i, y_{ji}), \quad y_{ij} = F_{ij}(X_i, U_i),$$
it is indicated with $E_i\left(X_i, (y_{ji})_{j=1}^{N_{SS}}\right)$ and $F_i(X_i, U_i)$ their vectorial form and with $(y_{ji})_{j=1}^{N_{SS}} \in \mathbb{R}^{n_{M_i}}$ the vector of inputs coming from the other disciplines.
- $G_{ji} : \mathbb{R}^{n_{X_i}} \times \mathbb{R}^{n_{X_j}} \times \mathbb{R}^{n_{U_j}} \to \mathbb{R}$: mapping to the inputs required for the analysis of the discipline D_i from the analysis of the discipline D_j. It is the composition of the two mappings $E_{ji} \circ F_{ji}$
$$M_{ji} = E_{ji}(X_i, y_{ji}) = E_{ji}(X_i, F_{ji}(X_j, U_j)) := G_{ji}(X_i, X_j, U_j).$$

In order to understand the different formulations of the optimization problem, the following definitions state

Definition 2.1.1. *Individual discipline feasibility* holds if all equations of an analysis code for a specific discipline fulfill
$$W_i(X_i, M_i, U_i) = 0, \ \forall i = 1, ..., N_{SS}.$$

Definition 2.1.2. *Multidisciplinary feasibility* holds if in addition to the individual disciplines feasibility, all outputs of each discipline exactly match the inputs of the others through the interdisciplinary mappings, and vice versa
$$W_i(X_i, M_i, U_i) = 0, \quad M_{ji} = G_{ji}(X_i, X_j, U_j) \ \forall i, j = 1, ..., N_{SS}.$$

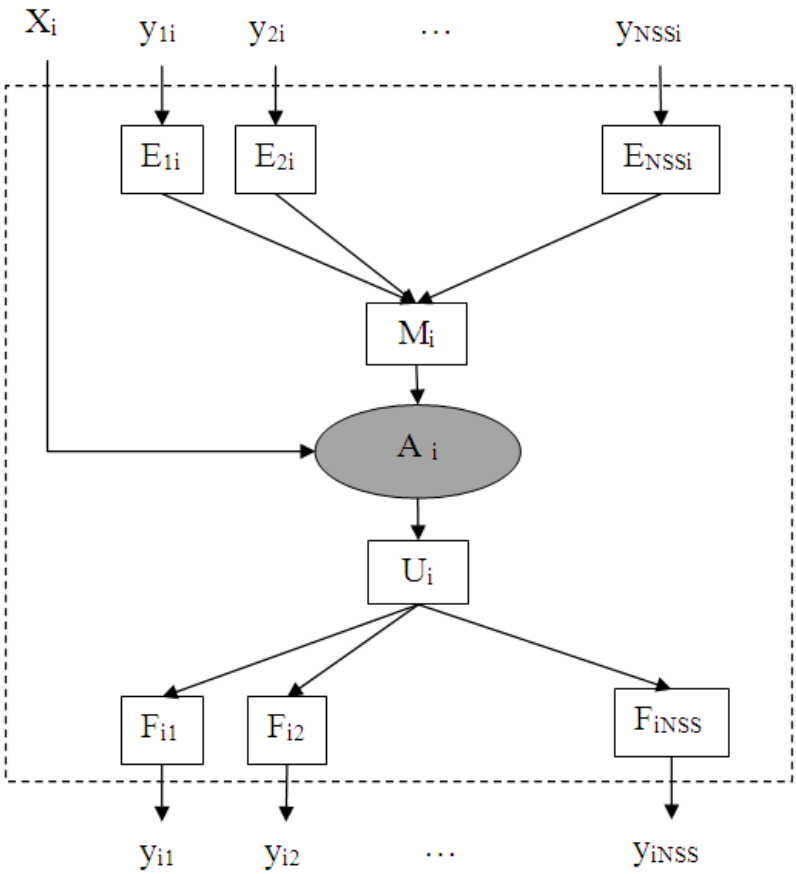


FIGURE 2.1. Disciplinary block.

Three different formulations of the multidisciplinary optimization problem can be identified. They differ on the kind of feasibility maintained during the iterations of the optimization process. In the Multi-Disciplinary Feasible (MDF) approach, complete multidisciplinary feasibility is required at every iteration, while in the Individual Disciplinary Feasible (IDF) and All At Once (AAO) it is expected only after the process convergences. The difference between the former approaches is that in IDF individual discipline feasibility is guaranteed at each iteration whereas in the AAO formulation this is not necessary. A similar classification with different naming conventions is proposed in [**12**] but it is not adopted here. Each formulation is analyzed in more detail in the following paragraphs, together with the mathematical statements of the derived optimization problems.

2.1.1. Multi-Disciplinary Feasible. The most straightforward definition of the multidisciplinary optimization problem is the MDF formulation. The optimization algorithm controls only the design variables and a complete Multidisciplinary Design Analysis (MDA) is computed for the corresponding values, maintaining the coupling of all disciplines. Hence full multidisciplinary

feasibility is ensured at each optimization iteration.
The problem can be formulated as

$$\begin{aligned} \min_{X \in \Omega} \quad & f(X, Y, P) \\ \text{subject to} \quad & c(X, Y, P) \leq 0, \end{aligned}$$

where $X \in \Omega \subseteq \mathbb{R}^{n_x}$ is the vector of design variables obtained from the disciplines design variables

$$X = \otimes_{i=1}^{N_{SS}} X_i,$$

Ω is the feasible design space defined by the design variables box constraints, $Y \in \mathbb{R}^{n_y}$ is the vector all the output values computed through the single discipline analysis,

$$Y = \otimes_{i=1}^{N_{SS}} Y_i = \otimes_{i,j=1}^{N_{SS}} y_{ij}$$

and $P \in \mathbb{R}^p$ is the parameter vector. Then $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_{\text{obj}}}$ is the objective function with $n_{\text{obj}} \geq 1$, for multiple objective cost functions. $c : \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \times \mathbb{R}^{n_p} \to \mathbb{R}^m$ is the constraints design function and n_x, n_y, n_p are the dimensions of the design variables vector, disciplines output and system parameters, respectively.
The MDF formulation has the advantage of limiting the number of variables to the minimum possible; however, a significant fraction of CPU time is spent in ensuring full multidisciplinary feasibility even far away from the optimum of the problem. In case of large and expensive problems the computational burden becomes too high, if the MDA requires iterations to converge the design.

2.1.2. Individual Disciplinary Feasible. The need for full multidisciplinary feasibility is eliminated and just individual disciplines feasibility is maintained at each iteration. The optimizer drives the individual disciplines toward multidisciplinary feasibility and optimality by controlling the interdisciplinary design data. The coupling variables in common among two or more disciplines are therefore duplicated, indicated with

$$\overline{Y} = \otimes_{i=1}^{N_{SS}} \overline{Y}_i.$$

Compatibility constraints are imposed to be satisfied at convergence of the optimization process. The i-th disciplinary analysis is evaluated in the doubled input variables $(\overline{y}_{ji})_{j=1}^{N_{SS}}$ and not in the values returned by the other disciplines. The outputs of the discipline D_i are computed as

$$Y_i = F_i\left(X_i, A\left(X_i, E_i\left(X_i, (\overline{y}_{ji})_{j=1}^{N_{SS}}\right)\right)\right).$$

The optimization problem is therefore enlarged in number of optimization variables and constraints, and can be formulated as

$$\begin{aligned} \min_{X \in \Omega, \overline{Y} \in \Theta} \quad & f(X, \overline{Y}, Y, P) \\ \text{subject to} \quad & c(X, \overline{Y}, Y, P) \leq 0 \\ & \overline{y}_{ji} = y_{ji}, \forall i, j = 1, ..., N_{SS}, \end{aligned}$$

where $\Theta \subseteq \mathbb{R}^{n_y}$ is the region of admissible values for the doubled coupling variables, f and c are defined as for the multidisciplinary feasibility case, with the difference that the coupling variables belonging to different subsystems are treated as different variables. The continuity constraints for the coupling variables of the discipline D_i is defined as

$$\overline{y}_{ji} = y_{ji} \quad \forall j = 1, ..., N_{SS}$$

this is equivalent to request the multidisciplinary feasibility condition $M_{ji} = G_{ji}(X_i, X_j, U_j)$, indeed

$$\begin{aligned} \overline{y}_{ji} &= E_{ji}^{-1}(X_i, M_{ji}) = E_{ji}^{-1}(X_i, G_{ji}(X_i, X_j, U_j)) = \\ &= E_{ji}^{-1}(X_i, E_{ji}(X_i, F_{ji}(X_j, U_j))) = E_{ji}^{-1}(X_i, E_{ij}(X_i, y_{ji})) = \\ &= y_{ji} \end{aligned}$$

The main advantage with respect to the MDF formulation lays in a higher computational efficiency, since expensive iterative MDA are no longer required even for coupled problems. It is also easier to modify or add single modules. The drawback is represented by the addition of variables and constraints to the problem, so that the IDF progressively loses efficiency as the number of coupling parameters among disciplines increases.

2.1.3. All At Once. The optimizer directly controls all design parameters and disciplinary outputs thus increasing greatly the number of optimization variables. At each model evaluation, rather than solving the set of analysis equations, their residual

$$\widehat{W}_i(X_i, (y_{ji})_{j=1}^{N_{SS}}, Y_i) \quad \forall i = 1, ..., N_{SS}$$

is evaluated, where $\widehat{W}_i = F_i \circ W_i \circ E_i$.
No kind of feasibility, neither individual nor multidisciplinary is guaranteed at each iteration, but they are both ensured after convergence. The analysis module therefore just evaluates the residuals of the governing equations, whereas the optimization algorithm assumes the difficult task of satisfying such equations through the additional compatibility constraints. The optimization problem can be formulated as follow

$$\begin{aligned} \min_{X \in \Omega, Y \in \Theta} \quad & f(X, Y, P) \\ \text{subject to} \quad & c(X, Y, P) \leq 0 \\ & \widehat{W}_i(X_i, (y_{ji})_{j=1}^{N_{SS}}, Y_i) = 0, \forall i = 1, ..., N_{SS} \end{aligned}$$

where $\Theta \subseteq \mathbb{R}^{n_y}$ is the region of admissible values for the disciplinary outputs, f and c are defined as for the two previous cases, with the difference that the introduced variables are included in the functions definition.
Even though according to [**11**] the AAO methodology may seem to be the most attractive in terms of computational efficiency, it requires a high degree of software integration; almost nothing can be reused of previously available disciplinary codes, greatly increasing the required work load, so that very few applications of this method are reported in literature.

	MDF	**IDF**	**AAO**
Feasibility at each iteration	Multidisciplinary feasibility	Individual discipline feasibility	None
Feasibility at convergence	Multidisciplinary feasibility	Multidisciplinary feasibility	Multidisciplinary feasibility
Optimization variables	Design variables	Design and coupling variables	All design and behavior variables
Optimization problem type	Small and dense	Average, depending on the coupling level	Large and sparse
Optimization efficiency	Low	Medium, depending on the coupling level	High
Optimization effectiveness	High, no design solutions can be lost in the optimization process	Medium, if coupling are neglected for efficiency, some solutions may be lost	High, no design solution may be lost in the optimization process
Reuse existing disciplinary codes	Medium, codes can be reused but different analysis need coupling	High, only set up of compatibility constraints is needed	Low, most of the code has to be rewritten
Maintainability	Medium	High	Low
Parallelization	Low	Medium	High

TABLE 2.1. MDO problem formulation comparison.

2.1.4. Comparison. Following the above considerations and with the objective of identifying the most suitable problem formulation for the applicative problem, the basic characteristics, advantages and disadvantages of the three approaches are summarized and compared in Table 2.1.
The IDF formulation has the higher level of maintainability and reusability of existing disciplinary analysis codes. The drawback of this approach lays in the optimization side, whose efficiency strictly depends on the number

of coupling variables and exploitation of the optimization problem sparsity structure. The potential to parallelization of IDF is related to the coupling relations between disciplines. If all the coupling input variables are doubled than all the subsystem analyses can be executed in parallel. Otherwise if a partial IDF formulation is used, breaking for example just the feedback links among disciplines almost no potential for parallelization exists.
The most straightforward MDF approach is well suitable for small low fidelity MDO problems. The level of maintainability of the disciplinary codes is lower with respect to the IDF approach because the coupling between disciplines needs to be explicitly implemented in a sequential evaluation of the disciplinary analysis modules, rendering a parallelization of the multidisciplinary analysis not possible. Hence the MDF approach is most appropriate for low level of analysis fidelity because the entire multidisciplinary analysis runs consecutively and within an optimization process small computational time is required. This is the reason of the low optimization efficiency level of the MDF approach that strictly depends on the involved type of analysis.
The strength of the AAO methodology is in the optimization efficiency and effectiveness. The optimizer does not waste time trying to achieve any kind of feasibility, neither individual nor multidisciplinary, when far from an optimum. Indeed, in AAO the subsystem analysis evaluates the residual of the analysis equation rather than solving the complete set of equations. Its main drawback lays in the level of reusability of the disciplinary codes and their maintainability. Since the single discipline feasibility is not assumed during the optimization iterations, the disciplinary analysis needs to be rewritten and readapted to the methodology, replacing the computation of the complete analysis code with the evaluation of the residuals.

2.2. MDO Decomposition methods

Problem decomposition methods are intended as the task of subdividing the multidisciplinary model into smaller blocks, grouping together highly coupled design variables, constraints and governing equations. The purpose is to reduce the complexity of the single analysis, intended as the coupling level with the other disciplines. Hence, the overall optimization process efficiency increases while still preserving the most important interdisciplinary interactions.
Problem decomposition methods can be classified in two main categories (see for example [**4**] for a thorough explanation of this classification): Hierachic Decomposition (HD) and Non Hierarchic Decomposition (NHD). The choice between these two approaches is strictly related to the coupling characteristics of the specific system under study. On the other hand, No Decomposition (ND) methods directly couple the model with any kind of optimization algorithm, and the model itself is just a black-box function taking as inputs the design variables and returning design objectives and constraints. The ND approach may result inadequate for applications involving a high number of design variables and constraints, or fully coupled problems which require

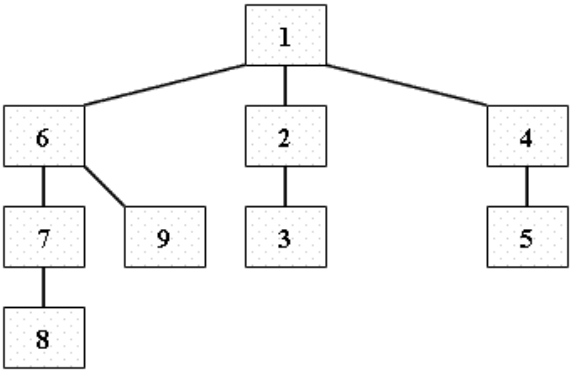
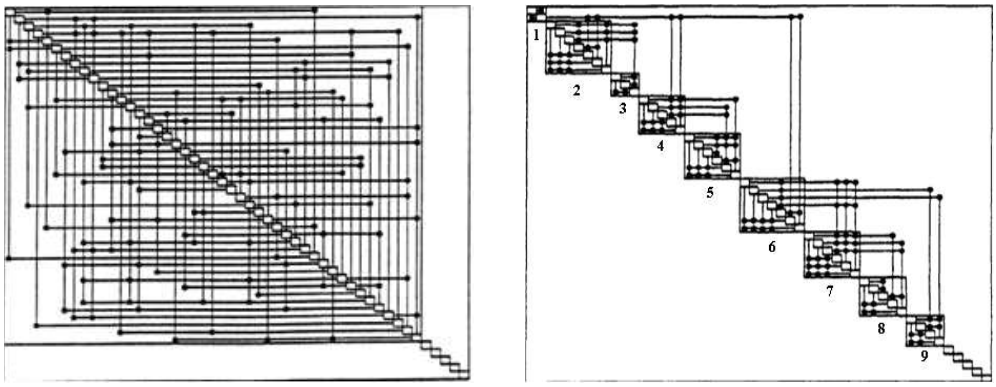


FIGURE 2.2. Example of hierarchically decomposed system.

several iterations to converge to a consistent design as well as long evaluation times.

2.2.1. Hierachic Decomposition. Hierachic Decomposition (HD) applies if the system can be divided into a set of modules forming a precise hierarchy, as the one shown in Figure 2.2. The boxes represent data converters that transform inputs into outputs, describing the physical subsystems or the analysis disciplines. The most important feature of hierarchically decomposable systems is that the flow of input and output information is only in the vertical direction and no data is transmitted between pairs of boxes located at the same decomposition level, as for example the boxes 6-2-4 of Figure 2.2. This implies that the entire task at hand is naturally separable into subtasks. This simplifies the process, since each block only needs the information coming from the higher levels, hence analyses and optimizations at the same level can be executed in parallel.
For those cases in which the subdivision of tasks is not straightforward, a formal method has been developed in which a set of candidate modules is identified on the basis of the designer experience and placed in a random order as square boxes on the diagonal of a diagram as shown in Figure 2.3, which is called N-square diagram. Note that feedforward and feedback information paths are represented respectively above and below the diagonal. The output information is flowing out from the vertical side of the white boxes while the input information is flowing into the horizontal side. The matching of the outputs of one discipline with the inputs of the other discipline is represented by black dots. Figure 2.3(A) represents a set of randomly grouped modules where no particular organization is visible. However, a HD can be identified by means of a formal procedure, described in [**13**], which systematically permutes rows and columns in the N-square diagram driven by rules incorporating the principle of not allowing any lateral exchange of data. The modules can thus be regrouped as in Figure 2.3(B), so that feedbacks have either been eliminated or limited to clusters that can now be seen as single



(A) Example of N-square diagram: system before re-ordering procedure

(B) Example of Nsquare diagram: system after re-ordering procedure

FIGURE 2.3. Nsquare diagrams

modules in a HD system, reducing the structure of the problem exactly to that represented in Figure 2.2.

2.2.2. Non Hierarchic Decomposition. A fully coupled multidisciplinary system cannot be decomposed into a purely hierarchic pyramid of modules, because no reshuffling of the modules in the N-square diagram can eliminate transmission links among those at the same level. For this particular case, there is no priority of placing one of the modules at a top level in a hierarchical structure. Thus, all modules must be treated at the same level, forming a one-level, not hierarchical, coupled system. Figure 2.4 shows the simplest example of three disciplines in a fully coupled system. The flow of information between the second and the third discipline is in the lateral direction. In such a case, the approach is called NHD, resulting in a more complex handling of the optimization problem, since the information has to be exchanged also in the lateral direction and no parallel execution of the disciplinary modules is achievable.

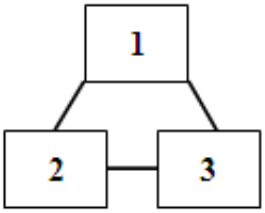


FIGURE 2.4. Example of no hierarchically decomposed system.

2.3. MDO Architectures

The concept of problem decomposition is a further classification of MDO architectures which concerns the structure of the multidisciplinary analysis and optimization, rather than the MDO problem formulation. Some of the most relevant MDO architectures that have been proposed in literature are presented in the following subsections, each classifiable both as ND, HD or NHD and as MDF, IDF or AAO. Note that all these architectures are presented for single-objective cost functions, whereas some considerations about their extension to generic multi-objective problems are given in Subsection 2.3.7. Besides, Subsection 2.3.8 summarizes all the presented techniques and draws some conclusions about the proposed methodologies, with the purpose of outlining a bibliographic comparison with respect to the main MDO goals stated at the beginning of the chapter.

2.3.1. Black-Box Optimization. The most straightforward idea of MDO is represented by a simple black-box approach: a single design model is built from the disciplinary analyses and the model is directly coupled with the optimizer, which recursively calls the model evaluation procedure with different values of the design variables, moving toward feasibility and optimality (Figure 2.5). The Black Box Optimization (BBO) method is a ND strategy with MDF problem formulation and a single System Level Optimization (SLO). Due to its extreme simplicity, it usually represents the first attempt in the development of any MDO code. The mathematical formulation of the system level optimization problem is the same as in Section 2.1.1:

$$\begin{aligned} \min_{X \in \Omega} \quad & f(X, Y, P) \\ \text{subject to} \quad & c(X, Y, P) \leq 0. \end{aligned}$$

The development of more detailed disciplinary models leads to the necessity of more refined MDO techniques. In fact, despites an extreme simplicity and easiness of integration with any local or global optimizer, BBO represents a "brute force" approach, whose efficiency can only be marginally improved by reordering the analysis sequence to minimize feedback coupling or temporarily suspending those couplings that appear weak at a particular design point. Besides, BBO can occasionally exhibit optimization convergence problems, due to the failure of the MDA process on the convergence of the internal iterative loops, and it presents almost no MDA parallelization potential.
Nevertheless, BBO still represents a good starting point for the development of any MDO environment for two main reasons: first, it is very well suited for relatively small problems that do not need iterative loops to close the design; and secondly, it can serve as a valuable mean of comparison for the design solutions obtained with more efficient approaches based on approximation, decomposition or non-MDF formulations, whose accuracy always has to be verified.
As a final note, other worth mentioning ND approaches are those employing

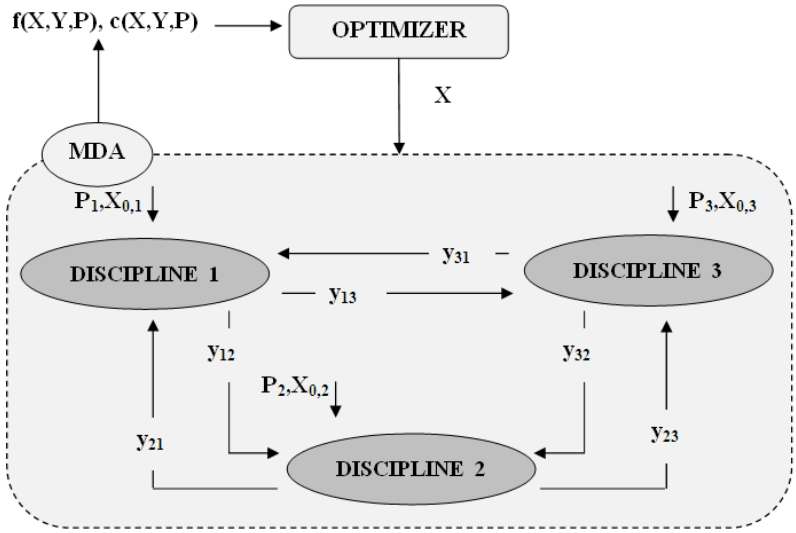


FIGURE 2.5. BBO architecture.

either IDF or AAO problem formulation together with a single SLO, in charge of the entire design authority as for BBO. Whereas AAO formulation has not found wide application due to the huge effort it requires for the integration of the disciplinary models, IDF-SLO methods have been successfully applied in several research areas and have been shown to provide significant computational savings with respect to BBO in two-way coupled problems [**14**]. The disadvantages of IDF lay in the increased number of variables and constraints and in the multidisciplinary feasibility being guaranteed only at optimization convergence, which can result in the lack of any consistent design information in case of not convergence of the optimization algorithm. The effectiveness of the IDF problem formulation has to be verified case by case.

2.3.2. Nested Optimization Loop. In some engineering applications, it is possible to identify disciplinary subspaces presenting very weak coupling with the system level problem in the bottom-up direction. In this case, the disciplinary blocks can be separated and considered at a lower level in a Nested Optimization Loop (NOL): the design variables coming from the rest of the MDA can be frozen by the system level optimizer, and the NOL variables can be optimized to achieve the desired objective within the disciplinary constraints coming from the higher level, as represented in Figure 2.6. Of course, more than one inner loops can be considered in a generic system. The system level optimization problem is formulated as in the BBO case, with the difference that every evaluation of the model requires an optimization subproblem to be solved, during the analysis of the i-th disciplinary

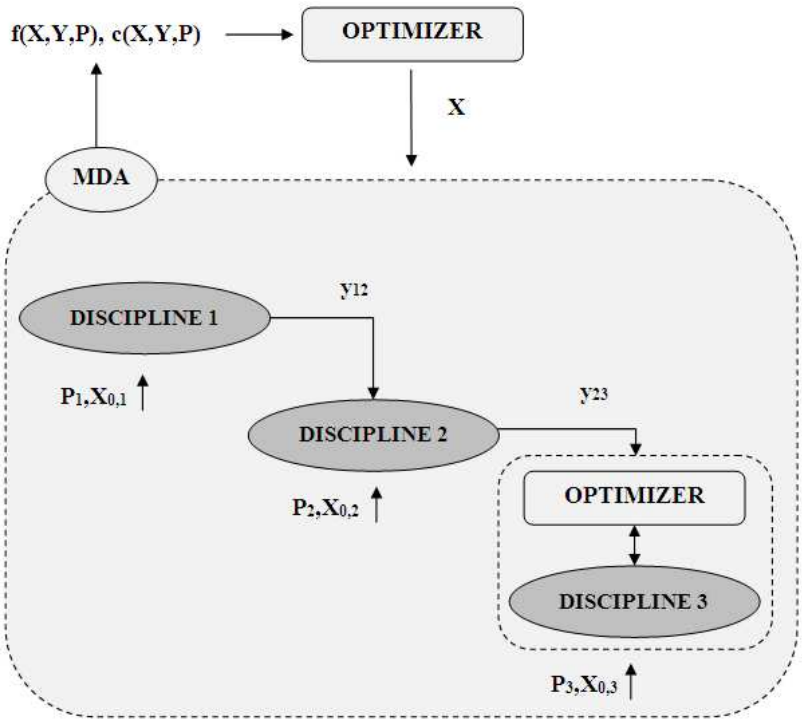


FIGURE 2.6. NOL architecture.

model, having the form

$$\begin{aligned} \min_{X_i \in \Omega_i} \quad & f_i(X_i) \\ \text{subject to} \quad & c_i(X_i) \leq 0 \end{aligned}$$

where X_i are the design variables of the i-th discipline, $\Omega_i \subseteq \mathbb{R}^{n_{x_i}}$ the design search space for the i-th discipline, $f_i : \Omega_i \to \mathbb{R}^{n_{\text{obj}}}$ and $c_i : \Omega_i \to \mathbb{R}^{n_m}$ are the restriction of the objective and constraints functions over the i-th subspace. All other design and coupling variables are fixed at the top-down analysis values.

Full multidisciplinary feasibility is maintained both in the outer and in the inner loops and no lateral communication is allowed between blocks at the lower level, therefore classifying this approach as MDF-HD. Moreover, no information can flow in the bottom-up direction, in order to avoid internal loops with the nested optimization problem, possibly leading to the loss of good solutions in case the disciplines in the nested loop have higher system-level effect than anticipated. This disadvantage has to be carefully weighted against the gain in optimization efficiency before deciding to proceed with the NOL approach, and a verification of the optimality of the achieved results should be performed at least for some representative cases. Finally, in terms of re-usability, maintainability and potential for parallelization, the NOL approach does not present particular advantages nor disadvantages with respect to BBO.

2.3.3. Multilevel Optimization by Linear Decomposition. Multilevel Optimization by Linear Decomposition (MOLD) is the first problem decomposition approach introduced in MDO, proposed by Sobieszczanski-Sobieski and other researchers at NASA LaRC's MOB back in 1982 [**15, 16**]. It is probably the most deeply investigated procedure based on multi-level decomposition and it can be applied to any kind of HD system. In MOLD, an iterative procedure is set up involving the successive analysis and optimization of the black-boxes at each level. Analysis information flows in the top-down direction by fixing the values of the higher level design variables, whereas optimization information is collected by means of sensitivity derivatives.
In every i-th subsystem, the following optimization problem needs to be solved, where $(y_{ji})_{j=1}^{N_{SS}} \in \Theta_i \subseteq \mathbb{R}^{n_{M_i}}$ are the coupling input variables of the discipline D_i:

$$\min_{(y_{ji})_{j=1}^{N_{SS}} \in \Theta_i} R_i = \min_{(y_{ji})_{j=1}^{N_{SS}} \in \Theta_i} \sum_{k=1}^{m} r_k \left((y_{ji})_{j=1}^{N_{SS}} \right)^2$$

subject to the box constraints on the disciplinary design variables and with the function $r_k : \Theta_i \to \mathbb{R}$ defined as

$$r_k \left((y_{ji})_{j=1}^{N_{SS}} \right) = \begin{cases} c_{k|i} \left((y_{ji})_{j=1}^{N_{SS}} \right), & \text{if } c_{k|i} \left((y_{ji})_{j=1}^{N_{SS}} \right) > 0 \\ 0, & \text{otherwise,} \end{cases}$$

where $c_{k|i} : \Theta_i \to \mathbb{R}$ represents the restriction of the k-th constraint on the i-th disciplinary subspace. Hence, r_k represents the violation of the k-th constraint in the i-th subsystem and R_i will be referred to as a cumulative constraints of the discipline D_i. The optimal value R_i^* and the optimal solution $(y_{ji}^*)_{j=1}^{N_{SS}}$ of the i-th subsystem optimization problem are analyzed for their sensitivity to the i-th disciplinary design variables X_i fixed in the subproblem to the value returned by the SLO. Its optimal value and solution are expressed by means of linear extrapolation in terms of finite increments of the design variables ΔX_i, and returned as a function to the top level optimizer (see [**15**] for further details). In general, the linear extrapolation of a real-valued function h with respect to the increment of the dependent variable x is defined as

$$h \simeq h(x_0) + \dot{h}(x_0)\Delta x$$

where x_0 is the extrapolation starting point. The extrapolated function will be indicated from now on as $\widetilde{h}$.
The system level optimization has the task to minimize the multidisciplinary optimization problem objective, subject to constraints on the sum of the subsystems cumulative constraints

$$\begin{array}{ll} \min\limits_{\Delta X \in \Omega} & f(X_0 + \Delta X, \widetilde{Y}^*, P) \\ \text{subject to} & g(\Delta X) \leq 0, \end{array}$$

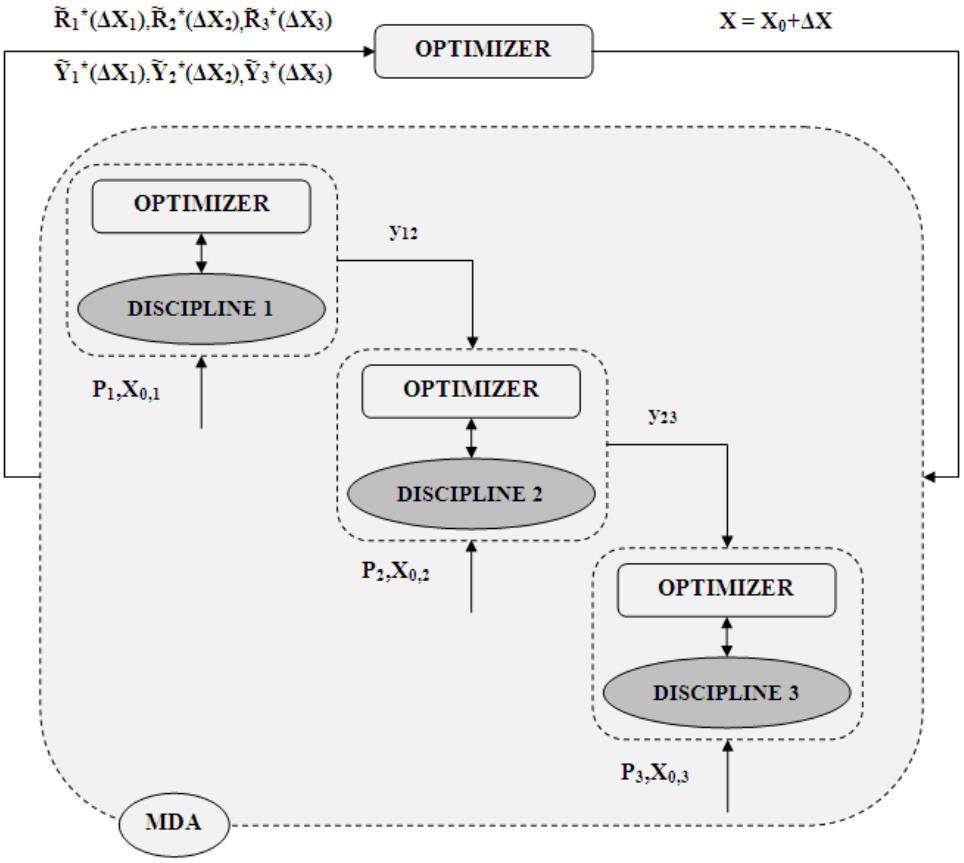


FIGURE 2.7. MOLD architecture.

where

$$g(\Delta X) = \sum_{i=1}^{N_{SS}} \tilde{R}_i^*(\Delta X_i)$$

is the sum of the linear extrapolation function of the optimal subsystem cumulative constraints and X_0 is the initial value of the design variables.
The described MDO architecture is represented in Figure 2.7 for a simple 3 level decomposition problem. This relatively simple procedure has the great advantage of allowing an efficient decomposition of complex and potentially very large systems in much smaller blocks, whose analysis and optimization can be executed in parallel if the blocks are located as the same level. Moreover, by exploiting the sensitivity information in the different subsystems, it allows to establish a rigorous mathematical link between design details and system level performances. For these reasons, MOLD has been shown to be extremely effective in dealing with hierarchically decomposable engineering systems. For example, the wing structure and aircraft performance optimization problem has been solved for a Lockheed L-1011-500 with a very high number of design variables (1303) and constraints (1950) in less than one day already with 1995 supercomputing technology as described in reference [**17**].

A more recent application of the method for the same problem is reported in [**18**], showing the possibility to extend the MOLD approach to problems with discrete variables, such as the cross-sectional stiffeners number and shapes. In this context, a genetic algorithm is exploited for the lower level optimization and coupling functions replace the sensitivity information for the integer variables.
Despite its remarkable efficiency, MOLD presents the important drawback of the complete inapplicability to any fully coupled engineering system. This limitation to hierarchically decomposable systems has led in the end of the 80's to the development of the first NHD approaches, which are presented in the next sections.

2.3.4. Concurrent SubSpace Optimization. Concurrent SubSpace Optimization (CSSO), proposed again by Sobieszczanski - Sobieski in 1989 [**19, 20**], is the first attempt to tackle non-hierarchically decomposable problems of large dimensions. CSSO stems from SubSpace Optimization (SSO) technique, in which the problem is divided in subspaces and each subspace optimizes a non empty subset of the design variables vector while holding the remaining constant. However, the traditional sequential SSO technique requires the expensive computation of the full system analysis for each subspace and does not allow concurrent execution of the separate blocks. CSSO aims at overcoming these obstacles by deeply exploiting the sensitivity information. The method, quite complex, has four initialization steps to be performed,

System analysis: the method begins with a full MDA starting from a trial vector of variables X_0, aimed at obtaining a multidisciplinary feasible first guess design solution. If there is a two-way-coupling between subsystems, the system analysis requires iterations for converging to a consistent solution.

Division in separate SSO: the vector X is partitioned into subsets to be used in the separate SSOs, for the single discipline analysis. Each variable must belong to a unique subset

$$X = \otimes_{i=1}^{N_{SS}} X_i, \qquad \mathcal{I}_{X_i} \cap \mathcal{I}_{X_j} = \emptyset, \;\; \forall i \neq j$$

The grouping of the design variables is than accomplished by judgment of the designer or through exploitation of the sensitivity information to rank the design variables in order of the degree of their influence on a particular disciplinary constraint and its contribution to the objective function.

System Sensitivity Analysis: a System Sensitivity Analysis (SSA) is performed to compute the derivatives of the behavior variables Y with respect to the design variables X. Each derivative is the measure of the influence of a particular design variable on the discipline output variables and they are used to construct a linear extrapolation of Y in terms of ΔX.

Definition of Cumulative Constraint (CC): for each i-th SSO a single CC is defined as a Kreisselmeier-Steinhauser (KS) function (first proposed by Kreisselmeier and Steinhauser in 1979, [**21**])

$$K_i(X_i) = \frac{1}{\rho} \ln \left(\sum_{k=1}^{m} \exp(\rho c_k(X_i, \widetilde{Y}, P)) \right)$$

where ρ is a factor controlled by the user. K_i represents with a single value the cumulative value of the constraint functions whose evaluation is restricted to the i-th subsystem analysis, considering constant the values of the design variables distributed to the other disciplines and obtaining their behavior variables values by linear extrapolation.

Knowing the system solution, the system sensitivity with respect to the design variables, the partition of the design variables over separated optimization processes in the different subsystems and knowing the cumulative value of the constraints in each analysis module, the optimization process can start. This is defined by three major steps:

SubSpace Optimizations: The SSOs are temporarily decoupled and executed concurrently. For each SSO the objective is to reduce the violation of its CC together with the minimization of the penalization of the system objective function. If the CC is already satisfied, then the system objective function is minimized as much as possible without violating that constraint. At the same time, since the optimization is confined to the single subsystem, the goal is also to reduce the cumulative constraint violation in the other subsystems and to avoid an arising violation of the already satisfied CC in other SSOs.

All this is achieved by using the System Sensitivity Derivatives (SSD) to account both for the cross-influence among subsystems and for the influence exerted on the system objective by those subsystem analyses that do not have a direct influence. For the i-th SSO, the objective is thus either to directly minimize the system cost function f, if it is computed within the i-th subsystem analysis, or to minimize a linear extrapolation of it, $\widetilde{f}(\Delta X)$, built with the system derivatives. N_{SS} constraints are then imposed in the i-th SSO, reproducing again with linear extrapolations the effect of the design variables X_i on the CC of the other SSOs. The possibility of further reducing the system objective at the cost of a slight violation of a CC, that may be then brought to satisfaction again by another SSO, is also allowed. The weighting among the CC of the N_{SS} subsystems and between their satisfaction or violation is governed by coefficients that are maintained constant during the SSO and are then used as optimization variables in the system level optimization (see [**19**] for

more details on such coefficients). The SSO problem can be mathematical formulated as

$$\begin{aligned} \min_{X_i \in \Omega_i} \quad & f_i(X_i) \\ \text{subject to} \quad & K_j(X_i) \leq K_{j,0} s_j (1 - r_{ij}) + (1 - s_j) t_{ij}, \\ & j = 1, ..., N_{SS} \end{aligned}$$

where $f_i(X_i)$ can be substituted by the linear extrapolation, if the i-th subsystem contributes to the minimization of the objective value only indirectly, through the changes induced in the other subsystems. $K_j(X_i)$ if $j \neq i$ is the linear extrapolation of the cumulative constraint of the j-th subsystem with respect to ΔX_i. Moreover, $K_{j,0}$ is the initial value of the CC, $s_j \in \{0, 1\}$ is a switch coefficient and $r_{ij} \in \mathbb{R}$, $t_{ij} \in \mathbb{R}$ are constant coefficients, optimized at system level, which allow violations of the constraints in other subsystems for an increasing value of the objective in the i-th subsystem.
Note that the double fidelity approach within the SSO (i.e. the i-th governing equations are accurately solved within the i-th SSO, whereas the effects of the other subsystem analyses are evaluated through the linear extrapolation) is the key aspect of the CSSO method enabling all participants to work in concert toward improving the design of the entire system while still remaining on the familiar mounds of their own specialty domain.

Optimum Sensitivity Analysis (OSA): The i-th SSO is executed for constant values of the coefficients r_{ij} and t_{ij} on a unique set of design variables X_i and for the same objective function f or its linear extrapolation. The minimum of the SSO is then depending on those weighting coefficients. For this purpose, it is possible to compute the derivative of f_i^* and K_i^* with respect to r_{ij} and t_{ij} for all $j = 1, ..., N_{SS}$ as shown in [**19**]. Those are used to build a linear extrapolation, which will be posed as objective of the system level optimization problem.

Coordination Optimization Problem (COP): in the system level COP, the objective is to seek new values of the weighting coefficients to further reduce the objective f, with constraints representing the normalization of the sum of the weights. Due to the linear extrapolation, the problem is reduced to a simple case of linear programming in the form

$$\begin{aligned} \min_{(\Delta r, \Delta t) \in \mathbb{R}^{N_{SS}} \times \mathbb{R}^{N_{SS}}} \quad & \widetilde{f}(\Delta r, \Delta t) \\ \text{subject to} \quad & \textstyle\sum_{j=1}^{N_{SS}} r_{ij} = 1, \qquad \forall i = 1, ..., N_{SS} \\ & \textstyle\sum_{j=1}^{N_{SS}} t_{ij} = 0, \qquad \forall i = 1, ..., N_{SS} \\ & 0 \leq r_{ij} \leq 1, \qquad \forall i, j = 1, ..., N_{SS}, \end{aligned}$$

where $r_{ij} = r_0 + \Delta r_{ij}$ and $t_{ij} = t_0 + \Delta t_{ij}$. Note how the coefficients represent the division of responsibility for the constraint violation

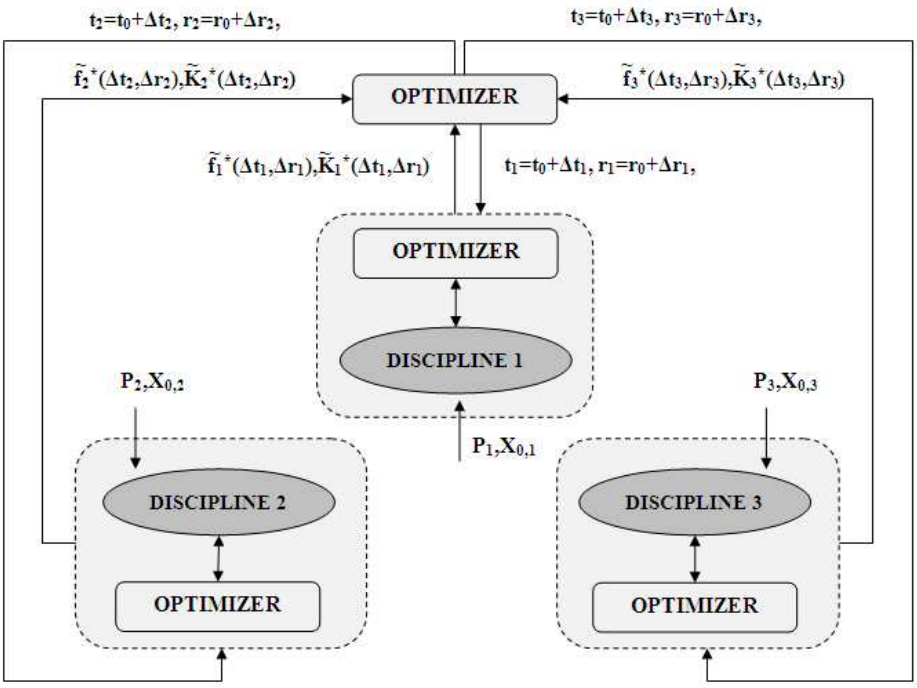


FIGURE 2.8. CSSO architecture.

reduction allocated to various SSO and that human intervention can be easily included in their definition.

The result of the COP execution is a new set of coefficients to be used in the next SSO. SSA, SSO, OSA and COP are repeated in an iterative fashion until some termination criterion is met after the COP. In Figure 2.8 a schematic representation of the architecture for a non-hierarchic design problem with three disciplines is shown. CSSO has been the first MDO architecture to be successfully applied to non-hierarchically decomposable problems [**20**].
In addition to being based on NHD, CSSO can be classified as an IDF formulation method: although the individual disciplines feasibility is maintained throughout the subsystem analysis performed within each SSO and the design variables are univocally divided among such SSOs, the outputs of the discipline D_i correspond only in first order approximation to the inputs of the other disciplines, thus full multidisciplinary feasibility is not ensured.
CSSO presents the great advantage of allowing an efficient NHD, thus avoiding the need of the expensive iterative multi-disciplinary analysis required with the MDF formulation. Moreover, all SSOs can be executed concurrently, allowing for a simple parallel implementation and for the exploitation of specialized methods both for sensitivities computation and for the optimizations. Finally, the high modularity of the method is an advantage for the maintainability of CSSO-derived software tools, and the human intervention is admissible to guide the procedure through the manual tuning of the

weighting coefficients.
The main drawback is instead represented by the linear approximation used to model the cross-correlation effects, which may lead to non-convergence of the method to a multidisciplinary feasible solution. Additionally, the entire method is based on the availability of sensitivity derivatives, preventing its use in applications involving discontinuities or discrete variables.
Partial solutions to these issues have been proposed in [**22**], where second order approximations are built in place of the linear extrapolations of the original method, and in [**23**], where Response Surface Analysis (RSA) obtained through artificial neural networks are used to provide information regarding the sensitivity derivatives in mixed continuous-discrete problems, where the discrete optimization is performed by a Simulated Annealing (SA) algorithm. However, such methods have shown the drawback of an increased computational effort with respect to the original CSSO implementation.

2.3.5. Bi-Level Integrated System Synthesis. The Bi-Level Integrated System Synthesis (BLISS) is an alternative MDO architecture applicable to generic non-hierarchically decomposable problems, proposed at NASA LaRC's MOB in 1998, see [**24, 25**]. Whereas CSSO is particularly suited for engineering systems presenting several coupled disciplines to be placed on the same level, BLISS is especially meant for those problems that include system-level design variables shared by one or more disciplines, for which the assignment to a single block is not physically meaningful. BLISS separates the SLO from the autonomous SSOs, by splitting the design variables vector X in two parts: first the SSO variables of discipline D_i, X_{l_i}, with $i = 1, ..., N_{SS}$ (local variables, not shared between subsystems); then the system-level variables $Z = \otimes_{i=1}^{N_{SS}} X_{s_i}$ which have to be shared by at least two subsystems

$$X = \left(\otimes_{i=1}^{N_{SS}} X_{l_i}\right) \otimes Z, \quad \mathcal{I}_{X_{l_i}} \cap \mathcal{I}_{X_{l_j}} = \emptyset, \forall i, j = 1, ..., N_{SS}.$$

The overall procedure can be summarized in the following steps:

System analysis: as usual, a full MDA is initially executed starting from the first guess values of X to compute the behavior variables Y, including all the disciplinary analyses and iterations until convergence.

System Sensitivity Analysis: a SSA is performed to compute the SSD of the behavior variables with respect to the design subsystem variables which are not shared among subsystems, with the objective of defining the linear extrapolation $\widetilde{Y}(\Delta X_{l_i})$ for the i-th module.

SubSpace Optimizations: in a first optimization step, a SSO is performed for each disciplinary block i, fixing Z and those X_{l_j} which do not belong to the i-th discipline. The cost function to be minimized is the linearized model approximation of the system objective, obtained through the SSA information of the previous step, subject to the

constraints computed within the i-th discipline

$$\begin{aligned} \min_{\Delta X_{l_i} \in \Omega_{l_i}} \quad & \widetilde{f}(\Delta X_{l_i}) \\ \text{subject to} \quad & c_i(\Delta X_{l_i}, \widetilde{Y}, P) \leq 0. \end{aligned}$$

By solving the SSO problems, the system objective improves through the reductions gained in each subdomain but maintaining feasibility of the constraints.

Optimum Sensitivity Analysis: for each disciplinary block i, the derivatives of the optimum solution found with respect to the variables kept constant are computed by means of an OSA, to construct the linear extrapolation of the objective with respect to the system level optimization variables Z, $\widetilde{f}_i^*(\Delta Z)$.

System Level Optimization: at a system level, a linear extrapolation of the objective function with the sensitivity information coming from the optimal values of the subsystem optimization problem defines the SLO problem

$$\min_{\Delta Z \in \Omega} \widetilde{f}(\Delta Z).$$

System analysis: a full MDA is repeated after each SLO, so that the system design achieved at each iteration is not only evaluated with linear approximations but also correctly analyzed according to the exact model. After this step, the iterative procedure is repeated starting from the SSA, unless some termination criterion is met.

A schematic representation of the presented MDO architecture is given in Figure 2.9. Note that whereas CSSO employs an IDF problem formulation, BLISS can be classified as MDF-NHD method, since at each iteration of the procedure an MDA is executed until convergence. As additional large difference with respect to the CSSO architecture, in BLISS the optimization procedures both at system and subsystem level are performed on the basis of the linear approximation of the objective function, and the high fidelity information is recovered only within the MDA performed after each SLO. Hence, the effectiveness of the method results strongly dependent on the non-linearity characteristics of the problem, as well as its efficiency depends on its coupling level, due to the increasing cost of the MDA iterations as the two-way couplings increases. Examples of successful applications of BLISS to the preliminary design of supersonic transports and to aeroelastic problems are reported in [**25, 26**].

An important role in recent applications of MDO is also played by an evolution of BLISS architecture, proposed in references [**27, 28**] again by researchers at LaRC, specifically targeted to massively parallel computing and named BLISS-2000. In this significantly different version, the costly SSA and OSA procedures are replaced by model fitting techniques (quadratic RSA) used to link the subsystem optimization data to the system optimization. The objective of each SSO becomes a sum of the subsystem outputs, each

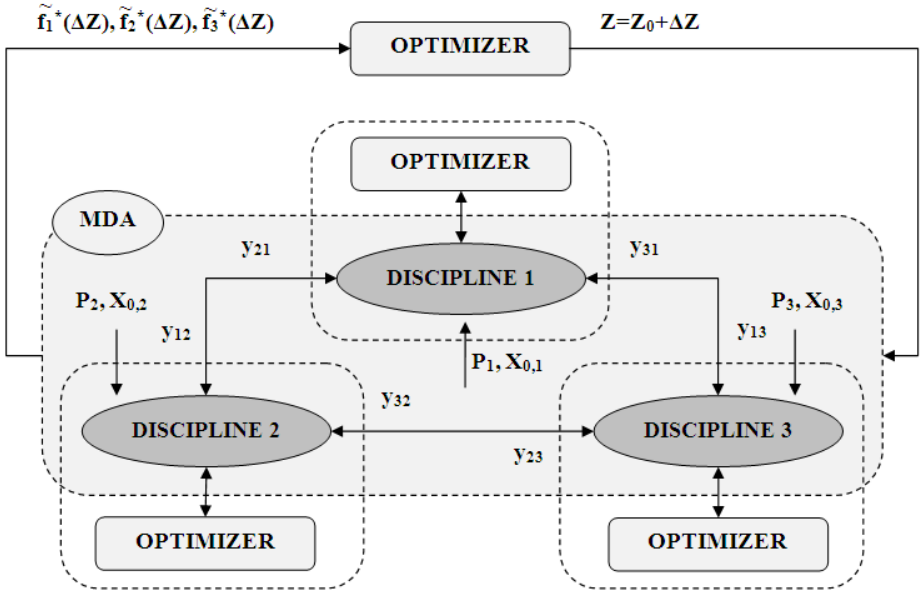


FIGURE 2.9. BLISS architecture.

weighted by a coefficient that is treated as a design variable in the system-level optimization. The response surfaces, to be used in the SLO procedure, are prepared by concurrent and potentially distributed operations performed within each discipline, hence scaling well with the increased number of processors. The elimination of SSA and OSA and the exploitation of Response Surface Methodology (RSM) (see [**29**] for details about the metamodeling technique) both leads to computational savings and allows a more straightforward application to non-smooth problems. This new method has been successfully verified for a supersonic business jet test case (see [**28**] for detailed results).

2.3.6. Collaborative Optimization. Collaborative Optimization (CO) is a MDO technique for both HD and NHD problems first proposed with the work of Kroo at Stanford University and Braun at NASA LaRC in 1994 [**30, 31, 33**]. CO is basically a bi-level optimization architecture based on the IDF formulation, where design variables shared by two or more disciplines are duplicated and compatibility constraints are imposed. When convergence is reached, all disciplines should agree on the value of the shared variables, which is established as target by the system-level coordination method.
Consider for instance an output parameter y_{ij} of a discipline D_i, is an input for the discipline D_j. In this case, an additional design variable is added representing a copy of y_{ij} used as input of the j-th module, and a compatibility constraint is imposed such that the output value computed in the i-th module shall correspond to the input value of the j-th module. The same

applies to shared design variables directly fed into two or more different modules, for which two or more copies are created and constraints are defined to enforce their consistency at optimization convergence. The input variables to a design discipline D_i are

$$X_{l_i} \otimes Z_i, \quad Z_i = (y_{ji})_{j=1}^{N_{SS}} \otimes X_{s_i}.$$

As already mentioned in Section 2.1.2, the IDF approach often increases the optimization efficiency, even when simply employing a single SLO. For example, it has been shown in [**30**] for a simplified aircraft design problem that the CPU times are reduced by 30% and the number of evaluations by 50% with respect to the MDF formulation. Note that with the fully decoupled system the additional advantage in terms of parallelism and modularity is to be traded against the growth of the number of coupling variables and constraints, that often excessively increases in practical problems involving many disciplines.

Although the IDF-SLO method presents important advantages, it still involves decomposition of the analysis only, whereas the actual design work is entirely done at system level; hence, subsystems are not permitted to apply discipline-specific design expertise and the use of specialized tools for disciplinary analysis and optimization is prevented. Especially for problems with weak interdisciplinary coupling and large numbers of domain-specific constraints, this centralized design approach is not appropriate, hence leading to the concept of CO, allowing to distribute both the analysis and the optimization among separate independent modules.

In CO, SSOs are in charge of all disciplinary design variables and are permitted to disagree with each other regarding the value of the shared parameters. The objective of the SSO of discipline D_i is to adjust the shared variables to minimize these interdisciplinary discrepancies, while satisfying the domain-specific constraints

$$\begin{aligned} \min_{X_i \in \Omega_i, (y_{ji})_{j=1}^{N_{SS}} \in \Theta_i} \quad & (Z_i - \bar{Z}_i)^2 \\ \text{subject to} \quad & c_i\left(X_i, (y_{ji})_{j=1}^{N_{SS}}, P\right) \leq 0, \end{aligned}$$

where $\bar{Z}_i$ are the outputs and design variables coming from the other disciplines. Let's define the auxiliary function $d_i(Z_i) := (Z_i - \bar{Z}_i)^2$.

The system level COP is responsible for ensuring that these discrepancies are made to vanish as the process reaches convergence, by specifying target values for the shared parameters, that therefore become system-level design variables. The mathematical formulation of the system level optimization problem is as follow:

$$\begin{aligned} \min_{Z \in \Omega_z} \quad & f(Z) \\ \text{subject to} \quad & d_i^*(Z_i) = 0, \ \forall i = 1, ..., N_{SS} \end{aligned}$$

Therefore, the goal of the SSOs is to match the system target values, to the extent permitted by local constraints, whereas the SLO aims at minimizing

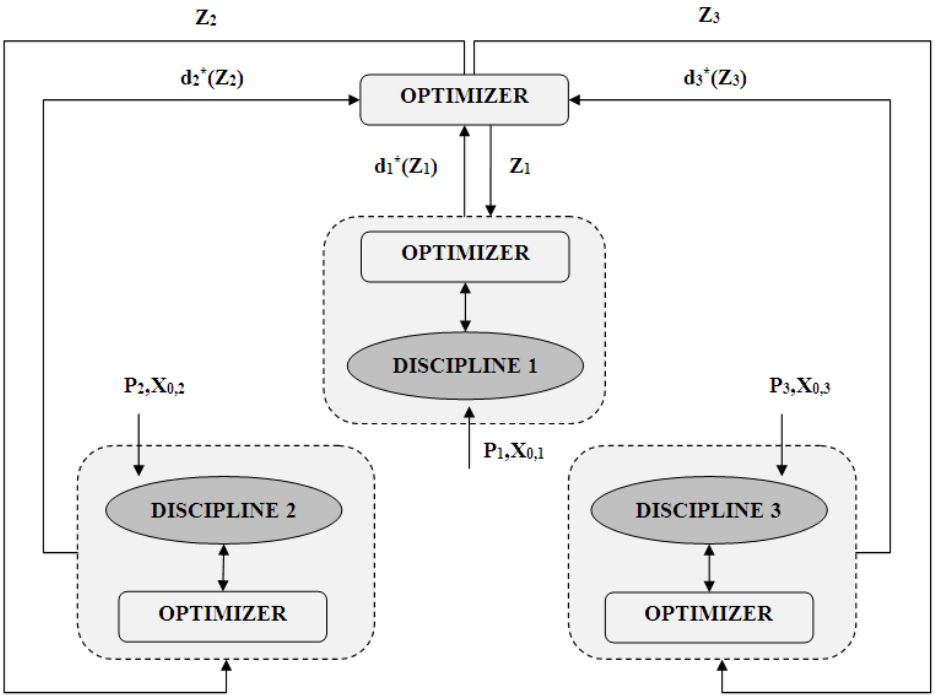


FIGURE 2.10. CO architecture.

the objective function while satisfying system-level compatibility constraints. Two-way communication between the SLO and the SSOs may occur at every iteration or at specified intervals in the optimization, depending on the problem characteristics and on the type of selected optimizer. Within this mechanism, the cross-correlation among the disciplinary groups is taken into account directly with the compatibility constraints, that reflect the influence of a given discipline on another through the system-level coordination.
Note that the CO design strategy is found in most design teams where a team leader (the SLO) is responsible for minimizing the overall objective while guiding a set of disciplinary experts (the SSO) into agreement. In Figure 2.10 a schematic representation of the two levels optimization architecture is shown. CO presents several commonalities with CSSO, the main difference being that while in both architectures each discipline attempts to satisfy its own constraints, the additional goal is to satisfy approximations of the other disciplines constraints in CSSO, whereas it is to match the target values on common variables in CO. Moreover, CSSO relies on linear or quadratic approximations, whereas CO is based on the exact disciplinary analysis.
In addition to the advantages in terms of efficiency stated above for IDF formulation, CO architecture presents the following interesting features:

- low interdisciplinary communication, since domain-specific variables, constraints and sensitivities need not to be shared;

- design authority is demanded to the different disciplinary areas, therefore possibly increasing the optimization efficiency;
- disciplinary problems can be addresses with domain-specific tools;
- even higher maintainability is achievable, since even significant modifications restricted to a discipline do not affect at all the communication flow among the SSO and the SLO;
- sensitivity information availability is not strictly necessary within CO. SSD may be anyway used to improve the overall efficiency, but no requirement is imposed on the continuity of the objective function and constraints or on the presence of discrete variables.

The main drawback of CO lays in an increase of the overall optimization problem size through the addition of an extra set of variables and constraints, which may result in a loss of efficiency in case of highly coupled problems for which the dimensions of such set are large. Several research works, such as those described in [**34, 35, 36, 37**], have been focused on the comparative analysis of efficiency issues in CO and other MDO techniques; as a general rule, algorithms based on decomposition or separability applied to truly coupled problems are less efficient than algorithms that directly work with the entire system. The former however becomes particularly interesting, also from the point of view of the overall efficiency, when the coupling among disciplines is not excessive.
Moreover, no theoretical proof of convergence of CO has been obtained, and practical difficulties have been encountered. For example, parameters which are common between two different subspaces are duplicated; each copy is then allowed to depart from the interdisciplinary target, hence being moved by the SSO toward the direction that is most suitable for its own domain. In the end, a small tolerance error is still present for both subsystems and if the preferred directions of the two subsystems are opposite, as it often occurs in engineering trade-offs, this errors sums up. As a result, a false improvement of the optimal objective function value with respect to the MDF formulation can be achieved, while not fully satisfying all of the constraints when the real target value is used in a complete MDA.
Despite these problems, CO architecture has been successfully applied in several research works; for example, reference [**30**] reports promising results for both a mathematical problem and a transport aircraft early conceptual design model, whereas a wide class of mathematical benchmarks are tested in [**34**]. Moreover, several applications of CO have emerged in different engineering areas; in the field of launch trajectories, references [**31, 33**] present the optimization of a lunar ascent decomposed in three arcs treated as disciplinary blocks, whereas a Single Stage to Orbit multidisciplinary model involving vehicle design, ascent trajectory and cost estimation relationships is solved within a CO environment in [**32**]. The integration in a CO architecture of high fidelity disciplinary codes has also been successfully tackled, as for example in [**38**] for a High Speed Civil Transport vehicle design involving high-order panel methods for subsonic and supersonic aerodynamics and structural finite

elements analysis.
Several enhancements of the CO architecture have been proposed over the years, including the exploitation of RSA and the application of Artificial Intelligence methodologies, see for example [**38, 39, 40**]. In particular, the combination of CO and RSM seems very promising, due to the low dimensionality of the subspaces and the possible exploitation of the problem knowledge given by the optimization process to improve the efficiency of the RSA fitting.
New variants of the method have also been recently proposed, most notably Modified Collaborative Optimization (MCO) by De Miguel in 2000 [**41, 42**] and Enhanced Collaborative Optimization (ECO) by Roth in 2008 [**43**], both at Stanford University. MCO uses penalty functions to include the compatibility constraints in the system objective, in order to have an unconstrained SLO procedure, and foresees additional variables to favor the compatibility constraints satisfaction. The key idea of ECO lays instead in the inclusion of models of the global objective and of all the subspace constraints in each subspace optimization problem, in order to communicate the sources of non-smoothness among disciplines while still maintaining the low dimensionality of the system level problem.

2.3.7. Extension to Multi-Objective Problems. Besides the multidisciplinary nature of most large-scale engineering problems, the additional challenge related to the existence of multiple contrasting objectives has to be considered for the studied application. Multi-objective optimization is an interesting field of study itself, hence the integration of multi-objective methods in multidisciplinary design architectures represent a critical research issue that needs to be addressed.
A variety of methods is today available to solve Multi Objective Problems (MOP), either heuristic-based, typically global optimization algorithms, or quantitative-based, which condense several objectives in a single cost function through weight coefficients. However, it is well known that the latter approach presents important drawbacks, including the inability to locate non-convex Pareto fronts and the overall inefficiency related to the necessity of running a complete optimization process for each desired Pareto-optimal solution. Refers to [**44**] for an extensive overview of quantitative-based methods for MOP in comparison with heuristic evolutionary techniques. On the contrary, heuristic global Multi-objective Optimization Algorithm (MOA), based on the concept of Pareto-dominance, consider in parallel an entire set of optimization variables vectors and hence directly provide the designer with a non-dominated front of Pareto optimal solutions. An extensive discussion about available MOA is given in Chapter 3.
Considering MDF problem formulations with SLO, the application of MOA to a black-box model returning the values of several objective functions is straightforward. No additional work with respect to the single objective case is required for the integration of optimizer and design analysis model.
The same can be said for HD approaches, such as MDF with NOL or MOLD.

However, the MOA can easily be applied only at system-level, for instance receiving the values of different cost functions from different disciplines, whereas the treatment of multiple objectives within the same discipline is more problematic because the information about a single optimal solution is foreseen to be communicated from the lower to the higher levels. Nevertheless, this is not an issue in many engineering problems, for which each subproblem represents a contrasting goal in the multidisciplinary design.
Regarding the NHD approaches instead, interesting solutions have been proposed in the past few years to modify available methods to include MOA both in SLO and SSO. Two of the most relevant multi-objective implementations of MDO techniques are shown in [**45, 46**], respectively proposing a Multi-Objective Pareto Concurrent SubSpace Optimization (MOPCSSO) and a Collaborative Optimization Strategy for Multi-Objective Systems (COSMOS). In MOPCSSO, the objective functions from other subspaces are introduced as constraints during the SSO, while OSA and COP are substituted by a new coordination procedure that considers multiple objectives. On the other hand, COSMOS modifies the system level coordination of CO in order to consider Pareto dominance criterion to merge the system and subsystem populations. In both cases, population based global optimization techniques are employed both in SLO and SSO to obtain sets of Pareto optimal solutions.
In conclusion, recent research has made it possible, even though not always straightforwardly, to extend the applicability of CDO techniques to MOP without losing the idea that each discipline should independently tackle its domain-specific goals and constraints, while at the same time moving toward interdisciplinary compatibility or optimality, guided by a system-level procedure.

2.3.8. Bibliographic Comparison Considerations. In the previous sections, several problem decompositions and MDO techniques have been described. Although a lot of research has been carried out in the last twenty years, leading to many interesting applications only partially covered in this section, a clear hierarchy among these methods has not been outlined. Yet such a hierarchy is unlikely to be ever defined, since the performance of each method is strictly related to the characteristics of the particular problem being solved, not to mention the practical implementation details varying from case to case.
Some attempts have been made to quantitatively compare different methods on the basis both of mathematical benchmarks or simple reference applicative problems (see for example [**47**] for a description of NASA LaRC's web-based MDO test suite). The most notable, mainly using as performance metrics the number of evaluations required to converge, the accuracy of the achieved solutions and the robustness of the approaches, are represented by the works of Balling [**48**], 1996, Alexandrov [**35, 37**], 2000, Tedford [**49, 50**], 2006 and Yi [**51**], 2008.

Note that all these studies are targeted to generic fully coupled MDO problems, hence do not include the HD methods presented. Note also that the provided results have not always been coherent, confirming the strong influence of problem characteristics and implementation details on the performances of CDO methods. Nevertheless, some general considerations can be evinced:

- Several comparisons [**48, 49, 50, 51**] among MDF, IDF and AAO methods involving only a SLO (i.e. ND approach), have shown an advantage in terms of computational efficiency of IDF over MDF, with a drastic reduction of the number of required function evaluations for a comparable effectiveness and accuracy. AAO is also shown to be at least as efficient and potentially better than IDF, but the complexity of its set-up complicates its application to practical cases.
- Again according to [**48, 49, 50, 51**], ND methods are generally more efficient than CSSO, BLISS and CO multilevel methods. However, the comparisons are run on the basis of benchmarks with few design variables, and it is likely that for larger problems the multilevel methods would show improved performance.
- Incoherent results are reported when comparing CSSO and CO, with a definite advantage of the former in [**51**] contrasting with its reported failure in [**48**]. The only comparison including BLISS, among those cited above, is instead that of reference [**51**], where such methods show better efficiency than both CSSO and CO.

In light of the mentioned quantitative comparisons and of the qualitative impressions obtained from the analyzed references, few conclusions have been drawn regarding the methodologies presented throughout the chapter. Table 2.2, 2.3 and 2.4 schematically summarize strengths and drawbacks of each architecture, covering all the desirable features listed at the beginning of the chapter.

2.4. Application of MDO Techniques to Launch Vehicle Design

The PRESTIGE research activity is focused on the development of a generic MDO development environment for Space Transportation Systems (STS). The selection of the most suitable MDO architecture is based on the current integrated models targeting the design of ELVs from a conceptual to an early preliminary level of detail. A brief description of the implemented multidisciplinary vehicle design is given in this section. The focus is more on the description of the interdisciplinary relations between disciplines rather than on the disciplinary models, in order to highlight the decomposition structure and coupling level of the system.
The models have been kept simple enough and no high fidelity models were introduced in the vehicle design to allow execution of a full MDA on a single processor in a computational time in the order of one second, trying to find a good compromise between model simplicity and accuracy. For details about the models selection, implementation and validation please refer to the joint

	Class	Effectiveness	Efficiency
BBO	MDF-ND	High all design variables are considered concurrently and no promising solution may be lost	Low for highly coupled system and CPU-intensive models
NOL	MDF-HD	Low, design solutions may be lost in neglecting the effect of the bottom-up coupling	Medium, may allow faster convergence to optimum than BBO
MOLD	MDF-HD	Medium, convergence is not ensured due to linear extrapolation in vertical direction (with the SLO)	High, the system is efficiently decomposed, no system level iterative analysis is needed
CSSO	IDF-NHD	Medium, convergence is not ensured due to linear extrapolation in horizontal and vertical direction (with the SSO)	High, no system level iterative analysis is needed and problem dimension is increased only by the N_{SS}^2 weight coefficients
BLISS	MDF-NHD	Medium, convergence is not ensured due to the complete problem linearization performed at each iteration	Medium/High linearized SSO and system optimization are fast but full MDA has to be performed at each step
CO	IDF-NHD	Medium, the system is not approximated but some couplings may be neglected for efficiency	Low/High iterations loops are removed and the optimization work is divided among SSO but efficiency strongly depend on the number of coupling variables

TABLE 2.2. MDO architecture comparison with respect to effectiveness and efficiency of the optimization process.

research work at Politecnico of Milano [**52**].
The Design Structure Matrix (DSM) of Figure 2.11 is a schematically representation of the disciplines involved in the MDA of ELV at a conceptual level, outlining its hierarchical structure and the interdisciplinary relations. The designed analysis includes the following disciplines: propulsion, geometry, aerodynamics, weights, trajectory, costs and reliability. Propulsion is the first SubSystem Module (SSM) to be called and allows to define, for each stage

	Class	Reusability & Maintainability	Parallelization potential
BBO	MDF-ND	Medium, disciplinary codes can be reused but their coupling need to be defined	Low, only inherently parallel portion of disciplinary codes can be distributed
NOL	MDF-HD	Medium, disciplinary codes can be reused but their coupling need to be defined	Low, only inherently parallel portions of disciplinary codes can be distributed
MOLD	MDF-HD	Medium, any disciplinary code can be reused but interdisciplinary communications integration is required	Medium/High, depending on the number of levels : all blocks at the same level can be executed in parallel
CSSO	IDF-NHD	Medium, any disciplinary code can be reused but interdisciplinary coefficients have to be included in the models	High, all blocks can be executed in parallel, only COP is sequential
BLISS	MDF-NHD	High, any disciplinary code can be reused just modifying SSA and OSA procedures	Medium, all blocks can be parallelized but system optimization and MDA must be executed sequentially
CO	IDF-NHD	High, disciplinary codes can be completely reused and interdisciplinary communications occur only through the SLO	High, all SSO can be executed in parallel, only the SLO has to be performed sequentially

TABLE 2.3. MDO architecture comparison with respect to reusability, maintainability and potential for parallelization.

and booster, engine components sizes ($y_{1,2}$) for the geometry analysis, inert masses ($y_{1,4}$) for weights estimation and performances ($y_{1,5}$) for the trajectory equations of motion integration. The propulsion discipline includes NASA's external public domain code Chemical Equilibrium with Applications (CEA), for theoretical rocket performance analysis. Then, geometry SSM exploits

	Class	Extension to discrete optimization	Notes
BBO	MDF-ND	High, any problem can be tackled provided that a suitable optimization algorithm is used	Low robustness due to possible not convergence of system analysis iterations
NOL	MDF-HD	High, any problem can be tackled provided that a suitable optimization algorithm is used	Low robustness due to possible not convergence of system analysis iterations
MOLD	MDF-HD	Medium, coupling functions have to be developed to replace SSD in case of discrete variables and discontinuities	Requires the problem to be hierarchically decomposable
CSSO	IDF-NHD	Low, SSD are strictly required	Especially suitable to systems decomposable in disciplines all at the same hierarchical level
BLISS	MDF-NHD	Low, SSD are strictly required	Especially suitable to systems showing system-level design variables in addition to disciplinary level variables at a lower hierarchical level
CO	IDF-NHD	High, any problem can be tackled with suitable optimization algorithms with no need of sensitivity informations	

TABLE 2.4. MDO architecture comparison with respect to the potential extendability to discrete optimization problems and further notes.

information from the system design variables and propulsion module to generate the geometry of the complete multistage vehicle. The two external tools included in the geometry SSM are not involved in the vehicle design cycle: 3-view and Silhouette are in fact public domain visualization tools which allow to plot respectively the 3 orthogonal and several perspective views of objects described in the geometry output files. The file generation script and these codes are not executed, for efficiency reason, in the MDA loops, but only at

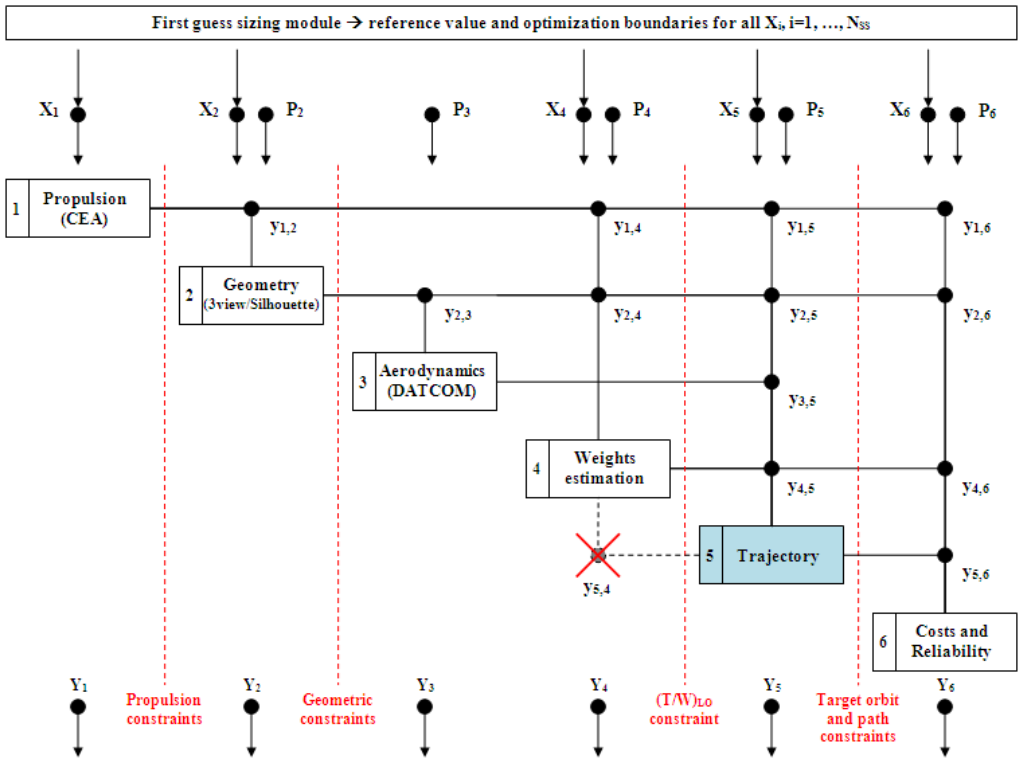


FIGURE 2.11. MDA for the conceptual design of ELV.

the end of the optimization, so that the final solutions can be visualized by the user. The aerodynamics and weight estimation SSMs follow and they may also be executed in parallel since no information are communicated between the modules. The former runs the external Missile DATCOM software for the estimation of lift, drag and pitching moment coefficients, with input values regarding the external geometry ($y_{2,3}$). The latter implements instead a number of Weight Estimation Relationships (WER) for all launcher components (structure, thermal, avionics), using propulsion weights ($y_{1,4}$) and external geometry data ($y_{2,4}$) as inputs. With all launcher design data now available, the trajectory simulation module is run, which allows defining whether the final target orbit can be reached and if the path constraints on the structural and thermal loads (dynamic pressure, axial acceleration, heat flux) are satisfied, with input values coming from all the above disciplines. Finally, the costs and reliability SSM is called to define additional objective functions related to the cost and reliability of the launch vehicle. Also the trajectory and the costs and reliability modules could be executed in parallel, guaranteeing that the information about the trajectory phases is provided to the cost module before the execution of the trajectory simulation.
An important remark about the MDA described above is the lack of iteration loops. For the concurrent optimization of launchers trajectory and design,

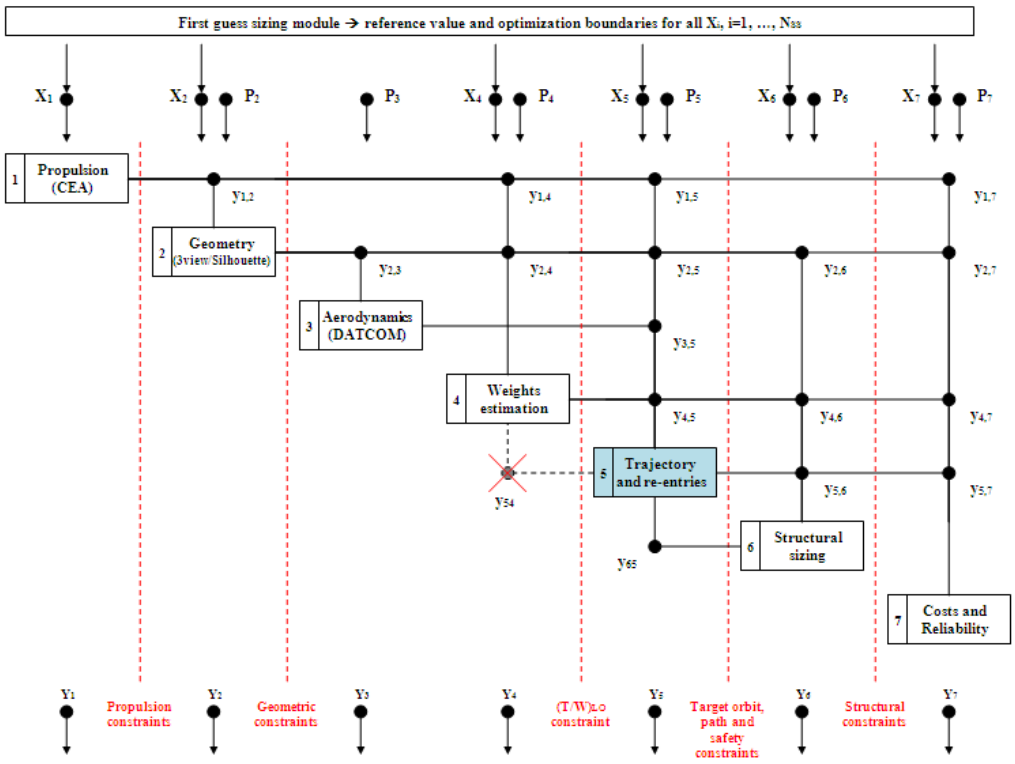


FIGURE 2.12. MDA for the early preliminary design of ELV.

one may expect the outputs from trajectory to be fed back to the weights estimation in terms of heat and structural loads. However, since simple WERs are used for the sizing of thermal protection system and main structure, these loads are reduced to the definition of the maximum values encountered during the ascent. These are the only three parameters that would have to be fed back, requiring iterations on the total dry mass to converge, as indicated by the dotted line below the diagonal in Figure 2.11. In order to avoid this iteration loop and the associated increase in computational time required for the MDA, the maximum values of the trajectory loads are included in the vector of the optimization variables, intended not as the largest actually encountered values but as maximum allowed values being the sizing inputs for weights estimation as well as constraints satisfaction thresholds for the trajectory integration. In this way, the optimizer is allowed to vary the level of structural and thermal loads, converging toward the optimal values as in a concurrent trajectory and design optimization.

The early preliminary MDO environment was obtained through upgrades which encompassed all disciplines. These improvements were derived in part from the critical analysis of the results of the detailed validation procedure, and in part from an independent review by ESA experts. In Figure 2.12 the updated DSM is shown. From a decomposition point of view, the major difference is in the introduction of a structural sizing discipline which is iterated

with the trajectory block to achieve convergence on the inert mass of fairing, stages and boosters. The structural module takes as input the design parameters from the geometry, weight and trajectory subsystems. In particular, the load values coming from the trajectory are used to size the structural components of the launcher, changing its inert mass and returning to the trajectory analysis. The optimization variables set enlarges with the addition of the structural design variables (X_6) and disciplinary related parameters of the enhanced propulsion and geometry analysis modules.
For the conceptual level ELV design, due to lower complexity of the model allowing a fast analysis and to the absence of internal cycles between disciplines, a simple BBO architecture has been selected with a global optimizer on top of the multidisciplinary analysis as first attempt to solve the MDO problem. This architecture has been classified as MDF-ND. A partial IDF formulation can rather be applied to the second version of the models introducing additional optimization variables for the inert masses given as input to the trajectory model and converging to the value returned by the structural analysis with the addition of compatibility constraints. Although IDF formulation has been reported in literature to be superior to the MDF formulation, this cannot be considered by generally valid. Its effectiveness depends mainly on the size of the additional variables and constraints, on the number of iterations required for the convergence of the inner loops, difficulty in achieving interdisciplinary compatibility and other features. On the given problem the number of iterations required to achieve convergence between the trajectory and structural model is limited (2 to 3 iterations for tolerance of 1% of the actual mass, see [**52**]) and the number of additional variables and constraints is equal to the number of the vehicle components. An a priori evaluation of the convenience of the MDF or IDF approach cannot be stated, a comparison of the two definitions on an applicative test problem is discussed in the chapter regarding the validation results. As opposed the All At Once (AAO) formulation is discarded a priori due to its inherent complexity in the model integration, the other more complex multi-level architectures, such as Multilevel Optimization by Linear Decomposition (MOLD), Bi-Level Integrated System Synthesis (BLISS) and Concurrent SubSpace Optimization (CSSO) are discarded due to the presence of discrete variables that require the exploitation of more advanced RSA to replace, where possible, the SSD need by the methods. For the fast model evaluation and limited number of optimization variables (restricted to a maximum number of 206 design parameters for the conceptual model and 364 for the early preliminary) the simple BBO architecture allows the process to reach convergence without the use of more complex techniques.
Due to the lack of feedback from the trajectory to the other SSMs, the trajectory optimization variables may be treated separately in light of their peculiar nature. A NOL is set up within the MDA process, with an entire trajectory optimization loop in place of the trajectory simulation. This would allow to exploit more efficient local, gradient based optimization algorithms for solving

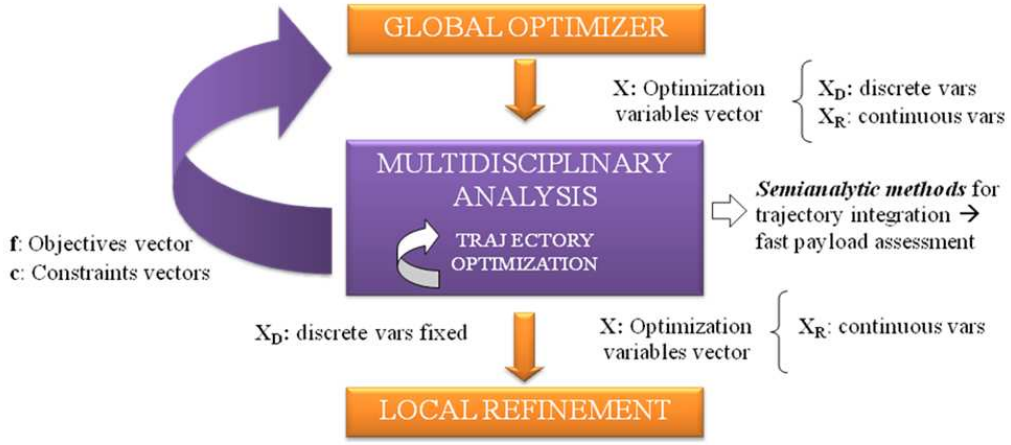


FIGURE 2.13. MDO for the conceptual and early preliminary design of ELV.

the trajectory optimization subproblem, involving just continuous variables and for which a reasonable first guess is available. This approach has also the advantage of reducing the SLO optimization variables set and number of constraints, accelerating the convergence of the top level global optimization. The disadvantage is that a complete trajectory optimization problem needs to be solved at each evaluation of the MDA also far from the system design optimal values. For this purpose, fast payload assessment optimization techniques based on the semi-analytic integration of the ascent trajectory have been investigated and integrated in the trajectory analysis module.
Suitable for both BBO and NOL architecture is the integration of local refinements of one or more of the obtained solutions, freezing all discrete design variables and selecting a single objective function from an aggregate of the possible objectives. The collaborative integration of a global strategy in the SLO to prune the wide design search space and capable of handling discrete variables, and the use of local gradient based techniques will efficiently improve the overall vehicle design. In Figure 2.13 is reported a schematic representation of both the described architectures. The continuous exchange of information between the MDA and the global optimizer on top of it, without exploitation of sensitivity information or decomposition methods, represents the execution of the multidisciplinary optimization as a BBO. Once the global optimization terminates, one or more of the achieved solutions can be further refined by fixing the discrete variables to their current values. The NOL optimization architecture is represented by the iteration loop over the trajectory module inside the multidisciplinary analysis, with the possibility to use numeric or semianalytic methods for integrating the rocket equations of motion.
The Collaborative Optimization (CO) architecture is an alternative to the presented approach. Since its strength relies in the parallelization of the multidisciplinary analysis through the decoupling of all interdisciplinary relations,

for the simplicity of the models introduced and for the fast analysis computation it is only foreseen for the extension of the MDA to higher fidelity models.

CHAPTER 3

MDO Problem

Contents

3.1.	**Global Optimization**	**55**
3.1.1.	Deterministic Strategies	57
3.1.2.	Stochastic Strategies	59
3.2.	**Multi-Objective Optimization**	**60**
3.3.	**Local Optimization**	**65**
3.3.1.	Optimality Conditions	66
3.3.2.	Optimization Algorithms	67
3.4.	**Application of Global and Local Optimization Strategies to MDO of Launch Vehicle**	**70**
3.4.1.	Global Single Objective Optimization Techniques	70
3.4.2.	Global Multi-Objective Optimization Techniques	76
3.4.3.	Local Optimization Techniques	85
3.4.4.	Reverse Communication	86

The mathematical formulation of the SLO problem in a BBO MDO architecture is the minimization (or maximization) of the objective function subject to system constraints on its set of design variables. For simplicity of the notation the dependency on the coupling variables Y and the vector parameter P is omitted in the objective and constraint function definitions. For $X \in \Omega \subseteq \mathbb{R}^{n_x}$

$$\begin{aligned} \min_{X \in \Omega} \quad & f(X) \\ \text{subject to} \quad & c(X) \leq 0, \end{aligned}$$

where $f : \Omega \to \mathbb{R}^{n_{\text{obj}}}$ is the objective function and $c : \Omega \to \mathbb{R}^m$ the constraints function.
The set of points satisfying the constraints is called the feasible region

$$D = \{X \in \Omega \mid c(X) \leq 0\}.$$

The problem can be rewritten as

$$\min_{X \in D} f(X).$$

Depending on the nature of the objective function and constraints (linear or nonlinear, single or multi-objective), of the search space (continuous or discrete) the optimization problems can be divided in different classes

Continuous or **Discrete**: the optimization variables belong to a feasible set that is a subset of the real space

$$X \in D \subseteq \mathbb{R}^{n_x}.$$

In some problems the variable X represents integer values (for example the number of thrusts in a rocket engine). Such problems are defined as **Integer Programming Problems** and the variables are in a feasible set such that

$$X \in D \subseteq \mathbb{Z}^{n_x}.$$

A subset of integer programming problems are the **Combinatorial Programming Problems** where

$$X \in D = \{0, 1\}^{n_x}.$$

If some of the variables in the problem are not restricted to be integer variables, the problem is called **Mixed Integer Programming Problem**

$$X = (X_r, X_d) \in D \subseteq \mathbb{R}^{n_r} \times \mathbb{Z}^{n_d}, \quad \text{with } n_r + n_d = n_x.$$

Constrained or **Unconstrained**: if there are no constraints on the design variables ($m = 0$) the problem is unconstrained. For constrained optimization instead $m > 0$. Unconstrained problems arise also as reformulations of constrained optimization problems, in which the constraints are added to the objective function as penalization terms.

Linear or **Nonlinear**: if the objective function and all the constraints are linear functions of X, the problem is called **Linear Programming** problem. Otherwise if some of the constraints or the objectives are nonlinear functions the problem is a **Non Linear Programming** problem.

Global or **Local**: many algorithms for nonlinear optimization problems find only a **local solution**, i.e. a point at which the objective function is smaller than all the other feasible points in a neighborhood. They do not always find the **global solution**, which is the point that has the lowest function value among all the points of the feasible region. Only for linear programming problems and convex programming problems the local solution is also the global one. An objective function that presents a big variety of local optima is called *multimodal* function.

Single or **Multi** objective: if the objective function is a scalar function, that is

$$n_{\text{obj}} = 1$$

the problem is said to be **Single objective**. In many engineering applications one is seeking a trade-off between different objectives, f is in this case a vectorial function with

$$n_{\text{obj}} > 1$$

and the problem is called a **Multi-objective** optimization problem. Multi-objective optimization problems can be transformed into single objective problems for example by means of aggregating functions, condensing all objectives in a single cost function with the use of weights coefficients.

Concerning the type of problem to be analyzed the focus moves toward global multi-objective optimization techniques for MINLP problems to tackle the SLO problem, and single objective local optimization methods for Non Linear Programming (NLP) to address the MDO local refinement problem and the trajectory optimization subproblem.

3.1. Global Optimization

The effectiveness of traditional local optimization techniques on multimodal objective functions strongly depends on the initial guess solution given to a method. If a previous knowledge of the problem is available, the designer can provide a good initial guess to the algorithm to ensure convergence to the global optimum. Otherwise the algorithm will mostly fail in the global search, getting trapped in one of the multiple local minima.
The purpose of global optimization is to find the best solution of a nonlinear optimization problem,

$$\min_{X \in D} f(X)$$

in the presence of multiple optima and a non smooth objective function.
Nevertheless, local optimization techniques will often play an important role also in global optimization strategies since some promising global approaches combine both global and local strategies of search. This is the case for example for Memetic Algorithms (MA) [**56**] that combine gradient based technique with evolutionary algorithms: the global search generates a set of trial points over the feasible region (solutions of the evolutionary strategy) and the local algorithm performs local descent search from the best available points, in an iterative loop that alternates the two steps till convergence. Obviously the best compromise between global and local strategies and the effectiveness of their use depend on the characteristic of the problem such as the geometry of the feasible region, the number of local minima and the sharpness of the objective function in the neighborhood of the global solution. However, the collaborative use of the global exploration capabilities of the first algorithm to prune the search space narrowing the area of search and the exploitation of local strategies to converge to the exact location of the global minimum is a very simple but effective approach that is foreseen also for the SLO MDO problem as local refinement of user selected global optimal solutions.
The limit of combining the two optimization approaches is the inefficiency of the local strategies in dealing with multiple objective problems and discrete variables. First a brief overview over the available global optimization algorithms and over the historical background that made them evolving to the actual state of the arts is given (see [**53, 54**] for a complete survey).

The methods that were first used in global optimization were deterministic techniques. They were introduced in the late 1950s with the advent of the first electronic computers into the research community. They are mostly based on the idea of trying to construct a sequence of approximate solutions which converge to the exact one by dividing the problem into smaller subproblems or approximations of the original one. One typical algorithm which embodies the previous technique is the Branch and Bound (BB) algorithm, 1960s [**58, 57**]. The basic idea is to recursively divide the feasible set (branching) and to move toward the global optimum region through resolution of intermediate optimization subproblems (bounding). It applies very well to cases where the optimization variables are discrete and a continuous relaxation of the optimization problem is admissible.
With the evolution of the computational power at the beginning of the 1990s different probabilistic global optimization approaches affirmed as new strategies. Among them it is worth to mention Simulated and Nested Annealing [**59, 60**], Tabu Search [**61**] and the large family of evolutionary strategies [**62**]. They are in general computationally less efficient than deterministic techniques, but due to their structure, they are able to tackle a wider range of problems and no assumptions on the model regularity and smoothness is required. For these reasons they are considered as one of the most promising techniques for solving global discrete, not relaxable non linear optimization problems.
There are several classifications of global optimization strategies. One is the already mentioned division between **deterministic** and **stochastic** algorithms. In the first category the model and the optimization variables are completely known and the algorithm performs through predefined steps. The stochastic component of the latter group instead lies either on the random sampling of the trial points, random parameters of the algorithm itself that made the single step not predictable, or on the use of a stochastic model for the objective function. Another division can be made between **exact** methods and **heuristic** methods. Exact methods provide a mathematical proof that the optimal solution can be found, while heuristic methods are not based on convergence theories. So no guarantee of finding the optimal solution is provided: the optimization process is constituted of iterative steps that improve the candidate solutions based on a measure of quality of their fitness, a function that combines indexes of optimality and feasibility.
An overview of the relevant methods is given below according to the first classification deterministic or stochastic with an internal differentiation between exact and heuristic methods. The objective of the section is to give a comprehensive overview over the available methodologies. For details about a specific algorithm please refer to the corresponding bibliography. The extension of the methodologies to the multi-objective case is discussed in the next section.

3.1.1. Deterministic Strategies. The first group are deterministic and exact global optimization strategies [**55**]. It means that no randomness is involved in the optimization process and the algorithm will always produce the same solutions for the same starting condition or initial state. The optimization steps are predictable and a proof of convergence exists.

Uniform grid search [**53**]: it is a trivial search strategy that makes use of a grid over the search domain to evaluate cost and constraints functions. The feasible point with the lowest objective value is taken as an estimate for the global minimum and a local improvement can be performed on the approximate global minimizer. The success of the local search obviously depends on the finesse of the search grid and global convergence can be trivially guaranteed by the fact that the mesh can be made arbitrarily dense. Such a simple scheme however rapidly becomes inefficient with the enlargement of the bounds on the optimization variables and the raising of the dimension. The computational load will increase as an exponential function of the dimensionality of the problem. Such a simple strategy, performed without the use of the local refinement, can be applied to every kind of optimization problems with continuous and discrete variables and no assumptions on the model regularities.

Complete (enumerative) search [**54**]: it is based on the simple principle of searching through all potentially optimum points in the search space, through enumeration of the possible candidates and evaluation of the objective. If for example the feasible region D is a polyhedra, and the objective function is concave, then it is possible to proof that the problem must have a global optimal solution which is a corner of D. Since D has a finite number of extreme points, the problem could be solved by enumerating the extreme points of D in an appropriate way until an optimal solution is found [**63**]. Enumerative methods have most applications in discrete and combinatorial problems and in concave programming. Convergence properties are trivially provable.

Homotopy and Trajectory methods [**65, 64**]: the two strategies have the ambitious objective of visiting all stationary points of the objective function on the feasible domain tracing the paths on the feasible space that include them. The solutions are then explored through enumeration techniques and evaluation of the objective. The two methods differ in the way of constructing their paths: the homotopy method makes use of homotopy transformations between the solution of a simplified problem and the original one, the trajectory problem solves a set of ordinary differential equations. The methodologies are applicable to smooth problems with continuous variables and the enumeration techniques employed guarantee convergence to the optimum.

Sequential approximation (relaxation) methods [**66**]: the idea is to build and solve a series of approximate (or relaxed) optimization

subproblems converging to the exact (or approximate) global optimum. A classification of such methods is based on the target of the approximation (relaxation), either specific model parameters or the entire system and subsystem models in a non-decomposed or decomposed problem, and the method employed to perform the approximated model fitting (RSM, Taguchi methods, Kriging [**29**]).The methods can be applied to a wide range of optimization problems with continuous and discrete variables and they are particularly suitable for expensive or noisy simulation models as a complete analysis is performed only in the experimental data points of the metamodeling techniques. The methods form a subset of the derivative-free optimization techniques, based on model approximation, as they are completely free from derivative computation or approximation. Method-specific convergence theories are available in the suggested reference.

Branch and Bound algorithm [**58, 57**]: the optimal solution is found by a successive adaptive partitioning of the feasible set. "Branch" refers to the partition process and "bound" to the lower bound for the optimal objective function value of each subproblem that is used to construct the next level of the branching tree. The technique is an implicit enumeration technique: all the solutions are investigated with a strategy to discard the points for which the non-optimality can be proved. The branch and bound methodology is applicable to pure integer or mixed integer programming problems and a definition of the continuous relaxation of the MINLP problem must apply. The implicit enumeration technique employed guarantees convergence to the optimum.

Interval arithmetic methods [**68**]: it is possible to develop a complete theory based on interval entities analogous to the real one. The strength of exploiting the global informations over large domains given by interval analysis in optimization methods ensure the convergence to all global optima. The idea is to start with an initial box and to delete the sub-boxes that cannot contain the global solution by a branch and bound procedure. The process terminates, when the bounds on the solutions and on the global minimum are below a predefined tolerance. The main drawback of the interval approach is its computational complexity. It is applicable to MINLP problems and non smooth functions.

On the other hand there is no proof of exactness for the following global deterministic strategy.

Sequential improvements of local optima [**67**]: the basic idea is to generate an improving sequence of local minima. Deflection techniques, tunneling and filled function methods are an examples of this approach. The tunneling method consists of two phases: seek for a local minimum, and apply a tunneling function to find a point in

the domain that has the same value of the objective function. The newly formed point is the starting point for the next iteration. The process terminates, when it is not possible to detect any point during the second phase. The last found local optimum is also the global one. There is no rigorously established convergence theory associated with these methods and they are applicable only to smooth continuous optimization problems.

3.1.2. Stochastic Strategies. Stochastic strategies are methods that contain not deterministic elements, either random generated algorithm parameters or stochastic approximations of model functions. As expected it is difficult to develop a rigorous convergence theory for such a class of algorithms, due to the randomness introduced in the optimization process. However two of them provide a convergence proof based on probabilistic theories and they can be classified as exact methods.

Random search methods [**69**]: the objective of these search methods is to find the global minimum with an adaptive-probabilistic distribution of random points over the feasible region. This algorithms ensure that the global minimum will be found with probability one as the sample size grows to infinity. The difference to the deterministic grid search algorithm lies in its adaptivity. The number of experimental points doesn't need to be decided in advance but it is generated in the successive steps. These methods are applicable to both discrete and continuous global optimization problems with very mild assumptions on the model regularity.

Random function approach [**70, 71**]: also known in literature as Bayesian methods, they are the stochastic counterpart of the sequential approximation approach with an adaptive probabilistic model for the approximation of the objective function. They are suitable for cost functions that have a highly computational load. They can deal with continuous and discrete variables and non smooth functions. A theoretical convergence to the global optimum is guaranteed only by generating a dense set of search points.

The larger group of stochastic optimization techniques are heuristic. They are most widely applied in practice but no mathematical proof of convergence exists.

Two-phase methods [**72**]: they are the stochastic counterpart of the deterministic grid search technique. They combine two phases of search: a global one and a local one. The process starts with a random sampling of the feasible space followed by the application of a local refinement. Multistart [**72**], Clustering methods [**73**] and Multilevel Single Linkage [**74**] are an example. The range of applications for the technique is constrained to the local search used. The greedy global strategy is suitable for both continuous and discrete variables with no assumptions on the model structure.

Simulated Annealing [**75**]: the technique is based on the analogy between minimizing a cost function and the cooling process of a material till it reaches its state of low energy equilibrium. The algorithm iteratively brings the actual state (optimization variables) to a lower level of the internal energy of the system (objective function). The changes between the states is done probabilistically. The new configuration is constructed by imposing a random displacement at each step. If the energy of the new state is lower than the previous one, the change is accepted. If the energy is greater, the new configuration is accepted with a probabilistic value. The probabilistic acceptance of upward moves is aiming to avoid the convergence to local minima. It is able to tackle global optimization problems with discrete and continuous variables under mild assumptions on the model regularity.

Evolutionary algorithms [**62**]: they are search algorithms inspired by the behavior of living organisms in the nature, based on the Darwinian principle of natural selection and survival of the fittest. A random set of solutions is generated initially. The individuals evolve through proper operators of the algorithm (as for example by recombination and mutation for the genetic algorithms) to create a new potential set of solutions. The potential solutions are then sorted on the base of their quality as solution, eventually recombined to the best solutions stored in the external archive. The process iterates evolving from the solution set of the best solutions found until a stopping condition is met. Particle Swarm Optimization [**76**], Genetic algorithms [**77**], Ant Colony Optimization [**78**], Differential Evolution [**79**] and Artificial Neural Networks [**80**] are the most representative techniques of this class. They have been proved to effectively solve a wide range of applied engineering problems, of particular interest are the most challenging cases, where the search space is large, discrete and the objective function is noisy, discontinuous or multimodal.

Tabu Search methods [**81, 61**]: it is a variant of the local descent method. The incombent solution explores the search domain with an adaptive memory that allows worsening moves (to avoid premature convergence to local minima) and forbids (tabu) for the next few steps to revisit points in the search space that have already been investigated. This technique was initially developed for integer programming problems.

3.2. Multi-Objective Optimization

The problem of optimizing concurrently two ore more objective functions falls into the category of Multi-objective optimization problems. In contrary to single objective optimization the purpose is not to find a unique global optimal solution but rather a set of solutions representing the compromise (trade-offs) between the different objectives.

Also in multi-objective optimization, as in single-objective, it is possible to

distinguish between local and global solutions: they will be referred as global frontier and local frontier.
The generic multiobjective optimization problem is defined as

$$\begin{array}{ll} \min\limits_{X\in\Omega} & f(X) \\ \text{subject to} & c(X) \leq 0 \end{array}$$

where X is the optimization variables vector, $f : \Omega \to \mathbb{R}^{n_{\text{obj}}}$, with $n_{\text{obj}} > 1$, the objective function and $c : \Omega \to \mathbb{R}^m$ the constraints function. As before the set of feasible points is denoted by D.
To extend the methodologies presented for global optimization to the multi-objective case, it is necessary to introduce some definitions.

Definition 3.2.1. A point $X_1 \in D$ *Pareto dominates* $X_2 \in D$ if

$$f_i(X_1) \leq f_i(X_2), \qquad \forall i = 1, ..., n_{\text{obj}}$$

and there is at least one component $j \in \{1, ..., n_{\text{obj}}\}$ such that

$$f_j(X_1) < f_j(X_2).$$

This is indicated by

$$X_1 \preceq X_2.$$

Definition 3.2.2. A point $X^* \in D$ is *Pareto optimal* if it isn't dominated by any $X \in D$,

$$X \preceq X^*.$$

In other words a solution is said to be Pareto optimal, or equivalently *non-dominated*, if there is no other point in the feasible space for which a decrease in one objective will not cause a simultaneous increase of at least one of the other objectives.

Definition 3.2.3. For a multiple objective optimization problem the *Pareto Optimal Set* is defined as

$$\mathcal{P}^* = \{X \in D \mid \nexists X' \in D \mid X' \preceq X\}.$$

Definition 3.2.4. The union of the objective values of all Pareto optimal points is called *Pareto front* or equivalently

$$\mathcal{PF}^* = \{f(X) \in \mathbb{R}^{n_{\text{obj}}} \mid X \in \mathcal{P}^*\}.$$

The Pareto front is the set of all solutions in the feasible space that are not dominated by any other possible solution. The minima in the sense of Pareto will lie on the boundary of the feasible region or in the tangent points of the objective functions. Generally it is not possible to derive analytically the equation of the front. Approximation techniques have been developed during the years to approach the Pareto frontier by successive iterations or to solve in parallel a sequence of single objective optimization problems.
A comprehensive survey of multi-objective optimization techniques is given in [**82, 83, 84**], the last two focusing mainly on global evolutionary multi-objective strategies. Evolutionary programming is the area of multi-objective

optimization research that in the last years registered the fastest growth. This is due to the intrinsic structure of the evolutionary algorithms, population based, well suited for an extension to multi-objective problems.
The multi-objective approaches are divided in methods that use the concept of Pareto dominance for the selection mechanism of the next iterates and methods that develop a special handling of the objective functions for reformulating the problem as single objective. The latter techniques are applicable to all the presented global optimization strategies while the former are typically for evolutionary algorithms.
The aggregation of the multiple objectives into a common single objective can be achieved by different techniques presented below, outlining their main advantages and disadvantages.

Weighted sum approach: the objectives are aggregated into a single function using weighting coefficients. The optimization problem becomes

$$\begin{array}{ll} \min\limits_{X\in\Omega} & \sum_{i=1}^{n_{\text{obj}}} w_i f_i(X) \\ \text{subject to} & c(X) \leq 0, \end{array}$$

where $w_i \geq 0$, and it is usually assumed that

$$\sum_{i=1}^{n_{\text{obj}}} w_i = 1.$$

By varying the values of the coefficients different solutions on the Pareto front are traced. To cover the entire front a sequence of single objective optimization problems needs to be solved, rendering the procedure very inefficient from a computational point of view. Moreover, this technique has the drawback of non generating proper Pareto optimal solutions in the presence of non-convex search spaces [**85**].

Goal Programming [**86**]: the designer has to assign targets to the objectives and the optimization problem is transformed in the problem of minimizing the sum of the deviations from the targets

$$\begin{array}{ll} \min\limits_{X\in\Omega} & \sum_{i=1}^{n_{obj}} |f_i(X) - T_i| \\ \text{subject to} & c(X) \leq 0. \end{array}$$

As prerequisite in the application of such a technique is a deep knowledge about the optimization problem to be able to assign meaningful target values to the objectives. The search space is explored by varying the T_i targets, and convergence to the Pareto front is achieved with a prior knowledge of the problem, to assign the targets close to the objectives values of the Pareto optimal points.

Goal attainment: it is a combination of the previous two techniques. Objectives goals are assigned as before, together with relative under or

over attainment weight coefficients. The problem becomes

$$\begin{aligned} \min_{X\in\Omega} \quad & \alpha \\ \text{subject to} \quad & c(X) \leq 0 \\ & f_i(X) \leq T_i + \alpha w_i, \;\; i = 1, ..., n_{\text{obj}}, \end{aligned}$$

where $\alpha \in \mathbb{R}$, and the weights $w_i \geq 0$ are normalized so that

$$\sum_{i=1}^{n_{obj}} w_i = 1.$$

It is possible to prove that the Pareto front can be covered varying the weight coefficients and the methodology is able to deal also with non convex problems [**87**].

The ε constraint method: the objectives are minimized one at a time, constraining the others below a certain level

$$\begin{aligned} \min_{X\in\Omega} \quad & f_j(X) \\ \text{subject to} \quad & c(X) \leq 0 \\ & f_i(X) \leq \varepsilon, \;\; i = 1, ..., n_{\text{obj}}, \; i \neq j. \end{aligned}$$

The main weaknesses of the approach are the same as listed above, computational efficiency, and a necessary a priori knowledge of the problem for covering the global Pareto front.

Lexicographic order: the objectives are sorted by user intervention. The optimization problem is divided in n_{obj} subproblems solved sequentially with a pre-established order and with additional constraints for not violating the satisfaction of the minimum values of the former subproblems. Assuming that $\{f_1(X), f_2(X), ..., f_{n_{\text{obj}}}(X)\}$ are the ordered objectives and f_i^* the minimum value achieved for the i-th objective. Then the i-th subproblem is defined as

$$\begin{aligned} \min_{X\in\Omega} \quad & f_i(X) \\ \text{subject to} \quad & c(X) \leq 0 \\ & f_j(X) = f_j^*, \;\; j = 1, ..., i-1. \end{aligned}$$

To cover the Pareto front different optimization runs with different sequences of objectives must be performed, heavily increasing the overall computational time.

Game theory: to each objective function a "player" is assigned. The player has the goal to minimize its objective. Assuming that the players are playing a non-cooperative game (i.e. the players make decisions independently) the intersection of the best strategy of each player is a *Nash equilibrium*, in the sense that no player can deviate unilaterally from this point for further improvement of the proper objective.

Weighted min-max approach: the deviations from the attained minima, in the n_{obj} single objective subproblems are estimated for the i-th

objective as

$$\bar{z}_i(X)\frac{|f_i(X)-f_i^*|}{|f_i^*|}, \quad \bar{\bar{z}}_i(X)\frac{|f_i(X)-f_i^*|}{|f_i(X)|}$$

assuming that the objective values do not vanish.
Defining $z_i(X) = \max\{\bar{z}_i(X), \bar{\bar{z}}_i(X)\}$, the desirable solution of the multi-objective problem is the one that gives the smallest values of all increments of all the objective functions

$$\min_{X\in D} \max_{i\in\mathcal{I}}\{z_i(X)\},$$

where $\mathcal{I}$ is the set of the objective indexes. The entire front can be covered by weighting the deviation function.

The exploitation of the concept of Pareto dominance in the evolutionary strategies led in the current years to the development of efficient multi-objective global optimization techniques. The particular structure of the algorithms, based on a family of solutions that evolves at each step, made the introduction of the concept of Pareto dominance in its ranking process possible [**77**]. The basic idea is to find a set of solutions that are Pareto non-dominated by the rest of the solutions of the feasible set, assign to them the highest rank and remove them from the group. The process then repeats recursively for lower values of the rank. This procedure can be applied for sorting the solutions of a current iteration and selecting a subgroup from it to apply the criteria of evolution of the species, resulting in a next generation of solutions that is different from the previous one and has an average better fitness.
Genetic algorithms is the larger class of evolutionary algorithms. They are divided in two groups:

First generation: they are characterized by the introduction of the concept of Pareto dominance in the process of selection of the population and for the niching operator to maintain the diversity and avoid premature convergence to local fronts. Representative algorithms of this class are Multi-objective Genetic Algorithm (MOGA) [**88**], Non Dominated Sorting Genetic Algorithm (NSGA) [**89**], Niched Pareto Genetic Algorithm (NPGA) [**90**].

Second generation: they exploit the concept of *elitism*. This means that they use an external archive to store the non-dominated solutions found in the previous generation in a way that the best solutions found in every iteration cannot be lost during successive iterations and a better global minima frontier can be achieved. The algorithms than differ in the way they interact with the external population. Representative algorithms of this class are Strength Pareto Evolutionary Algorithm (SPEA) [**91, 92**], Non Dominated Sorting Genetic Algorithm V.2 (NSGA2) [**93**], Pareto Archived Evolution Strategy (PAES) [**94**], Pareto Envelope-based Selection Algorithm (PESA) [**95, 96**], Micro Genetic Algorithm (Micro-GA) [**97, 98**].

Another group of evolutionary algorithms not classifiable as genetic algorithms already mentioned in the previous section get inspired by natural phenomena such as the colling state of a metal or the behaviour of an ant colony in the search of food. A corresponding reformulation of the already presented algorithms is avaliable for multiobjective optimization problems. Namely they are: Multi-Objective Simulating Annealing (MOSA) [**99**], Multi-Objective Particle Swarm Optimization (MOPSO) [**100**] and Multi Objective Ant Colony Optimization (MOACO) [**101**].

3.3. Local Optimization

Local optimization algorithms are exact methods that guarantee the convergence to the local optimum in a neighborhood of search. They are the most investigated optimization techniques and have their roots in the calculus of variations and the work of Euler and Lagrange. The development of linear programming falls back to the 1940s and it is the base of the modern optimization theory that rapidly grows and was developed in the last 70 years.
As already defined in the previous section, the general optimization problem is defined as

$$\min_{X \in D} f(X)$$

where $D = \{X \in \Omega \mid c(X) \leq 0\}$, $f : \Omega \to \mathbb{R}$ and $c : \Omega \to \mathbb{R}^m$ are sufficiently smooth functions. It must be pointed out that local optimization techniques restrict their field of application to single objective optimization problems with continuous variables. To extend the use to multi-objective optimization problems one of the aggregate techniques presented above must be taken into consideration.
Before introducing the optimality results some definitions need to be stated.

Definition 3.3.1. The real function $\mathcal{L} : \Omega \times \mathbb{R}^m \to \mathbb{R}$ defined as

$$\mathcal{L}(X, \lambda) = f(X) - \lambda^T c(X)$$

is the *Lagrangian* and the coefficients $\lambda \in \mathbb{R}^m$ are called *Lagrange multipliers*.

Definition 3.3.2. Given a point X in the feasible region, the *active set* $\mathcal{A}(X)$ is defined as

$$\mathcal{A}(X) = \{i \in \mathcal{I} \mid c_i(X) = 0\},$$

where $\mathcal{I} = \{1, ..., m\}$ is the index set of the constraint.

Definition 3.3.3. The Linear Independence Constraint Qualification (LICQ) holds if the set of active constraint gradients $\{\nabla c_i(X), i \in \mathcal{A}(X)\}$ is linearly independent, that is

$$\text{rank}(\nabla c_i(X), i \in \mathcal{A}(X)) = |\mathcal{A}|.$$

Note that if this condition holds, none of the active constraint gradients can be zero.

3.3.1. Optimality Conditions. These definitions allow the statement of the following optimality conditions (refer to [**102**], for a proof of the Theorems).

THEOREM 3.3.1 (First-order necessary condition). *Suppose that X^* is a local solution of the constrained NLP problem and that the LICQ holds at X^*. Then a Lagrange multiplier vector λ^* exists such that the following conditions are satisfied at the point (X^*, λ^*)*

$$\nabla_x \mathcal{L}(X^*, \lambda^*) = 0, \tag{3.3.1}$$

$$c(X^*) \leq 0, \tag{3.3.2}$$

$$\lambda^* \geq 0, \tag{3.3.3}$$

$$(\lambda^*)^T c(X^*) = 0. \tag{3.3.4}$$

These conditions are known as the Karush Kuhn Tucker (KKT) conditions.

REMARK 3.3.1. The last condition implies that the Lagrange multipliers corresponding to inactive inequality constraints are zero, hence it is possible to rewrite the first equation as

$$0 = \nabla_x \mathcal{L}(X^*, \lambda^*) = \nabla f(X^*) - \sum_{i \in \mathcal{A}(X^*)} \lambda_i^* \nabla c_i(X^*).$$

The optimality condition presented above gives information on how the derivatives of objective and constraints are related at the minimum point X^*. Another fundamental first order necessary condition, that gives additional information on the gradient of the objective function in the optimal point, can be stated. For this an additional definition is needed.

DEFINITION 3.3.4. Given a feasible point $X \in D$, a sequence $\{X_k\}_{k=0}^{\infty}$ with $X_k \in \Omega$ is a *feasible sequence* if, for all $k \in \mathbb{N}$, $X_k \in D \backslash \{X^*\}$ and

$$\lim_{k \to \infty} X_k = X.$$

Given a feasible sequence, the set of the limiting directions $w \in \Omega \backslash \{0\}$

$$\lim_{k \to \infty} \frac{X_k - X}{\|X_k - X\|} = \frac{w}{\|w\|}$$

is called the *Cone of the Feasible Directions*, $C(X)$.

Moving along any vector of this cone (with vertex in a local minimum point X^*) either increases the objective value or keeps it the same.

THEOREM 3.3.2 (First-order necessary condition). *If X^* is a local solution of the optimization problem and f is differentiable in X^*, then*

$$\nabla f(X^*) \cdot w \geq 0 \quad \forall w \in C(X^*).$$

For the directions w for which $\nabla f(X^*) \cdot w = 0$, it is not possible to determine, from first derivative information alone, whether a moving along this direction will increase or decrease the objective function. It is necessary to examine the second derivatives of the objective function and constraints to see whether this

extra information resolves the issue. The directions for which the behavior of f is not clear from the first derivative form the following set:

DEFINITION 3.3.5. Given a pair (X^*, λ^*) satisfying the KKT conditions
$C(\lambda^*) = \{w \in C(X^*) \mid \nabla c_i(X^*) \cdot w = 0, \text{ for all } i \in \mathcal{A}(X^*) \cap \mathcal{I}, \text{ with } \lambda_i^* > 0\}$
is called the *Critical Cone*.

Indeed for $w \in C(\lambda^*)$ from the first KKT condition it follows that

$$\begin{aligned} \nabla f(X^*) \cdot w &= \sum_{i \in \mathcal{A}(X^*)} \lambda_i^* \nabla c_i(X^*) \cdot w \\ &= 0. \end{aligned}$$

If X^* is a local solution, then the curvature of the Lagrangian along directions in $C(\lambda^*)$ must be non negative in case of qualified constraints. A positive curvature is instead a sufficient condition for a local optimum.

THEOREM 3.3.3 (Second-order necessary condition). *Let f and c be twice continuously differentiable, X^* is a local solution of the constrained problem and that the LICQ condition is satisfied. Let $\lambda^* \in \mathbb{R}^m$ be the Lagrange multiplier for which the pair (X^*, λ^*) satisfies the KKT conditions. Then*

$$w^T \nabla_{xx}^2 \mathcal{L}(X^*, \lambda^*)\, w \geq 0, \quad \forall w \in C(\lambda^*)$$

THEOREM 3.3.4 (Second-order sufficient condition). *Let f and c be twice continuously differentiable, X^* is a feasible point, $\lambda^* \in \mathbb{R}^m$ such that (X^*, λ^*) satisfies the KKT conditions and*

$$w^T \nabla_{xx}^2 \mathcal{L}(X^*, \lambda^*)\, w > 0, \quad \forall w \in C(\lambda^*), w \neq 0.$$

Then X^* is a strict local minimum of the constrained problem.

3.3.2. Optimization Algorithms. In the last 50 years a variety of approaches have been developed to solve NLP problems, first tackling the most simple unconstrained NLP problem and then expanding their application also to the constrained case. A starting point, denoted by X_0 is always provided to the algorithm by the knowledge of the user or left to the optimizer. The optimization process iterates exploiting information on the objective, constraints, their derivatives and the previous iterates to terminate whenever no further progress can be made or the optimal solution is approximated with acceptable accuracy.
The algorithm for unconstrained NLP are presented first. They are divided into two groups: line search and trust region.

Line search: the algorithm determines a search direction p_k and searches along this direction from the current iterate X_k for a new iterate with a lower function value. The step length to move along p_k can be found by approximately solving the minimization problem

$$\min_{\alpha > 0} f(X_k + \alpha p_k).$$

At the new point, a new search direction and step length are computed, and the process is repeated until convergence.

Trust region: the algorithm constructs a model function m_k whose behavior near the current iterate x_k is similar to that of the actual objective function f. The iteration direction of search p is found as solution of the problem

$$\min_{p \in \Omega} m_k(X_k + p),$$

where $X_k + p$ lies inside the trust region. If the solution does not produce a sufficient decrease in f, it means that the trust region is too large. In this case the trust region is shrunk and the minimization problem is solved again. Usually the trust region is the ball

$$\|p\| \leq \Delta, \quad \text{where } \Delta \text{ is the trust region radius}$$

and the model m_k is usually a quadratic function of the form

$$m_k(X_k + p) = f(X_k) + p^T \nabla f(X_k) + \frac{1}{2} p^T H(X_k) p$$

where H is the Hessian matrix of the Lagrangian.

The two approaches differ in the way they choose the direction and the distance of the move: Line Search fixes the direction p_k and optimizes the length of the step. Thrust Region instead first chooses the maximum distance of the move, the trust region radius, and than seeks for the best move to attain the best improvement of the objective function.

As example for line search methods there are (refer to [**102**] for the details about the methods):

Steepest Descent method, it chooses as search direction the descent one $p_k^{SD} = -\nabla f(X_k)$;

Newton methods, the search direction is the solution of the Newton equation $p_k^N = -H(X_k)^{-1} \nabla f(X_k)$;

Quasi Newton methods, they don't require the computation of the second order derivatives but use an approximation of it, noted as B, $p_k^{QN} = -B(X_k)^{-1} \nabla f(X_k)$;

Nonlinear Conjugate Gradient methods where the search direction is defined as $p_k^{CG} = -\nabla f(X_k) + \beta_k p_{k-1}$ with $\beta_k \in \mathbb{R}$.

Newton and Quasi-Newton methods are the ones that attain a super linear rate of convergence but they require the computation and the storage of the Hessian matrix. On the other hand methods that rely just on the gradient information are slower at convergence.

Most of the methods have a counterpart for the Thrust Region approach. In the quadratic model the Hessian matrix is substituted by the one used by each method (identity matrix for the steepest descent, H_k for the Newton method and its approximation B_k for quasi Newton methods). It is possible to prove that the resulting search direction is defined as in the Line Search methods and its length constrained by the Trust Region radius.

The presentation of the algorithms for the unconstrained case was necessary to introduce the techniques for solving constrained NLP problems as parts of them rely on the idea of converging to the solution of the constrained problem by approximating it with a sequence of unconstrained problems.
The algorithms for constrained NLP problems can be grouped in:

Penalty, barrier, augmented Lagrangian methods and **sequential linearly constrained methods**: they solve a sequence of simpler subproblems (unconstrained or with simple linearized constraints) related to the original one. The solutions of the subproblems converge to the solution of the primal one either in a finite number of steps or at the limit.

Newton-like methods: they try to find a point satisfying the necessary conditions of optimality (KKT conditions in general). The Sequential Quadratic Programming (SQP) method is part of this class.

The *Penalty methods* combine the objective function and constraints into a penalty function $\alpha(X)$ which is null for feasible points and positive otherwise. The problem to be minimized is the unconstrained problem

$$\min_{X\in\Omega} f(X) + \mu\alpha(X)$$

for a series of increasing values of the penalty parameter μ, such that $\mu\alpha(X) \to 0$ as $\mu \to \infty$, until the solution of the constrained optimization problem is identified with sufficient accuracy. From a computational point of view, super linear convergence rates might be achieved, in principle, by applying Newton's method to solve the minimization problem (or its variants such as Quasi-Newton methods). The algorithmic behavior is strictly related to the choice of the penalty parameter. If μ is large more importance is given to the feasibility than the optimality and the iterates could move to feasible regions far from the optimum, causing slow convergence and premature termination.
The *Barrier methods* or *Interior-Point methods* add terms to the objective function that act as a barrier and prevent the iterates from leaving the feasible region. For example, in the case of inequality constrained problems, a barrier problem can be formulated as

$$\min_{X\in\Omega} \theta(\mu),$$

where $\mu \geq 0$ and $\theta(\mu) = \inf\{f(X) + \mu b(X) : c_i(X) < 0, \forall i \in \mathcal{I}\}$. The barrier function b should be non negative and continuous on the feasible region and go to infinity as the boundary is approached from the interior. This would guarantee that the iterates do not leave the domain. The starting point must be chosen in the interior of the feasible region and Newton or Quasi-Newton methods can solve the successive barrier problem.
In the *Augmented Lagrangian methods* a penalty functions is added to the Lagrangian:

$$\mathcal{L}_A(X, \lambda, \mu) = f(X) - \lambda^T c(X) + \frac{1}{2\mu}\|c(X)\|_2^2$$

Fixing λ to some estimate of the optimal Lagrange multipliers and $\mu > 0$ to some positive value, it is possible to find a value of X that approximately minimizes $\mathcal{L}_A(\cdot, \lambda, \mu)$. Then the process is repeated updating λ and μ with the information from the previous X-iterate.
In *Sequential Linearly Constrained methods* at every iteration a Lagrangian is minimized subject to a linearization of the constraints.
Sequential Quadratic Programming has instead a completely different approach. It employs Newton-like methods to solve directly the KKT conditions of the original problem. The problem turns out to be a minimization problem of a quadratic approximation of the Lagrangian subject to a linear approximation of the constraints. The search direction p_k at the iterate (X_k, λ_k) is the solution of the problem

$$\begin{aligned} \min_{p} \quad & \tfrac{1}{2} p^T \nabla^2_{xx} \mathcal{L}(X_k, \lambda_k) p + \nabla f(X_k) \cdot p \\ \text{s.t.} \quad & \nabla c_i(X_k) \cdot p + c_i(X_k) \leq 0, \quad i \in \mathcal{I}. \end{aligned}$$

A trust region constraint can be added to the algorithm to control the length of the step and a Quasi-Newton approximation of the Hessian can be used instead of the second derivatives of the Lagrangian.

3.4. Application of Global and Local Optimization Strategies to MDO of Launch Vehicle

Among all the presented local and global optimization techniques, two single objective and five multiple objectives optimization strategies have been compared with representative algorithms from the class of deterministic and stochastic techniques. The algorithms are presented here in more detail and in the Chapter 5 a quantitative comparison of their performances on analytic and applicative test problems are given.

3.4.1. Global Single Objective Optimization Techniques. To solve the MINLP MDO system level optimization problem several global optimization strategies have been investigated. On the base of their success in engineering applications evolutionary strategies for single and multi-objective have been preferred among other stochastic-heuristic techniques. The stochastic approach is compared with two exact deterministic algorithms: the branch and bound algorithm nested with a local SQP technique to tackle mixed discrete and continuous optimization problems, for single objective and a derivative free optimizer of the class of sequential approximation techniques for bi-objective optimization problems.
The single objective strategies presented here are the Particle Swarm Optimization (PSO) and the Branch and Bound and WORHP algorithm (BBWORHP). While for the evolutionary strategy there exists the multi-objective counterpart, the BBWORHP algorithm can be extended to solve multiple objectives problems with one of the techniques presented in Section 3.2 with the main limitations already outlined. The weighted sum approach has been implemented in the definition of the MDO SLO problem. However, even if they

could tackle multiple objectives the two algorithms have been selected to efficiently solve just single objective optimization problems, and their performances will be validated on this class of problems.

3.4.1.1. *Particle Swarm Optimization (PSO).* The algorithm belongs to the class of evolutionary strategies, but it differs from the other techniques as it doesn't apply mutation or evolutionary operators to the incumbent solutions. The solutions are moved in the search space, modifying their position and velocity, toward their own previous best positions, and toward the previous best position attained by any member of the whole swarm. The process wants to recreate what in nature is observed as social behavior of groups of organisms.
The original algorithm is attributed to Kennedy and Eberhart [**76**]. In the following years several variants have been proposed for extending the methodology to multiple objectives [**100**] and expedients to improve the overall efficiency of the optimization process for continuous and mixed continuous-discrete variables [**103**].
The i-th particle in the search space is represented by the position vector

$$X_i = (x_1, ..., x_n) \in \Omega \subseteq \mathbb{R}^n.$$

The swarm is the union of all the particles and the best particle, i.e. the particle with the smallest function value, is indicated with index g

$$g = \{i \in \mathcal{I}_s \mid \min_i(f(X_i))\},$$

where $\mathcal{I}_s$ is the set of the particles indexes. The best previous position of the i-th particle is stored in the memory and indicated as $P_i = (p_1, ..., p_n) \in \Omega \subseteq \mathbb{R}^n$, while the current particle velocity is $V_i = (v_1, ..., v_n) \in \mathbb{R}^n$. The i-th particle moves, from the $k-th$ position and velocity to the $k+1$, within the search space, according to the following equations

$$\begin{aligned} V_i^{k+1} &= \omega V_i^k + k_1 r_{i1}^k (P_i^k - X_i^k) + k_2 r_{i2}^k (P_g - X_i^k), \\ X_i^{k+1} &= X_i^k + \chi V_i^{n+1}. \end{aligned}$$

ω is the inertia weight, acting on the velocity value, it is the trade-off between global and local search: high values increases the exploration capability of the algorithm (global search) but slowing down the convergence to the optimum while small values tend to facilitate local exploration and so the fine tuning of the current search area. A good choice for the inertial parameter is to start with high values to prune the search space and decrease it while approaching the global optimum. k_1 and k_2 are two positive constants, called the cognitive and social parameter. They represent the confidence that the particles gives to the information coming from the previous iterations of the particle itself and the global information coming from the swarm. r_{i1} and r_{i2} are two random numbers uniformly distributed in $[0, 1]$. χ is a constriction factor used to control and constrict the velocities and prevent their explosion. The tuning of these parameters is problem-dependent, however a lot of research has been

accomplished in this direction for finding some guidelines for properly setting up the algorithm [**104, 105**].
In general, the performance of each particle is measured according to a fitness function, which is the objective function penalized by the corresponding constraints violation

$$\widetilde{f}(X) = f(X) + \mu^T \cdot c(X),$$

where for $i = 1, ..., m$

$$\mu_i = \begin{cases} 0 & c_i(X) \leq 0 \\ \infty & \text{otherwise.} \end{cases}$$

$\widetilde{f}$ is used from now on as quality measure of the solution rather than the objective and constraint functions. The algorithm can be summarized in the following steps, where $l \in \mathbb{R}^n$ and $u \in \mathbb{R}^n$ are the optimization variables box constraints:

Algorithm 3.4.1. *PSO*

(1) *Initialization:*
- *Random initialization of the swarm* $X_i^0 \leftarrow\!\shortmid U(l, u)$, $\forall i \in \mathcal{I}_s$.
- *Initialization of the best particle's known position* $P_i \leftarrow\!\shortmid X_i$, $\forall i \in \mathcal{I}_s$.
- *Update the best known swarm position* $P_g \leftarrow\!\shortmid \min_i \widetilde{f}(X_i^0)$.
- *Initialize the particle velocity to zero* $V_i^0 = 0$.

(2) *Iteration* $k = 0, ..., N_{\max}$*:*
- *Compute particle's velocity* $V_i^k \leftarrow\!\shortmid V_i^{k+1}$, $\forall i \in \mathcal{I}_s$.
- *Update particle's position* $X_i^k \leftarrow\!\shortmid X_i^{k+1}$, $\forall i \in \mathcal{I}_s$.
- *Update the particle's best known position if* $\widetilde{f}(X_i^{k+1}) < \widetilde{f}(X_i^k)$, $\forall i$
$$P_i \leftarrow\!\shortmid \widetilde{f}(X_i^{k+1}).$$
- *Update the best known swarm position if* $\widetilde{f}(X_i^{k+1}) < P_g$
$$P_g \leftarrow\!\shortmid \widetilde{f}(X_i^{k+1}).$$
- *Iterate until stopping condition is met.*

(3) *Termination:* P_g *contains the best function value.*

The algorithm terminates either if the maximum number of iterations is reached or if for a given number of generations (N) the standard deviation of the normalized average best objective values of the entire set of particles (n_s) falls below a given threshold (ε_{tol})

$$\sqrt{\frac{1}{N-1}\sum_{j=1}^{N}\left(\frac{\overline{P}^j}{P_g^j} - \overline{P}\right)^2} < \varepsilon_{\text{tol}}$$

where $\overline{P}^j$ is the mean value of the best objective values attained by the particles at the j-th iteration, P_g^j is the best swarm objective value at the

j-th iteration and $\overline{P}$ is the mean value of the ratio

$$\overline{P} = \frac{1}{N} \sum_{j=1}^{N} \frac{\overline{P}^j}{P_g^j}.$$

3.4.1.2. *Branch and Bound and WORHP algorithm (BBWORHP).* The BB technique is well suited for solving Integer Programming problems. The methodology has its root in the work of Land and Doig, [**106**] 1960, for Linear Integer Programming (LIP) and it developed during the years for NLP applications by means of dividing the original problem into subproblems to be solved with convex programming or linear approximations [**57**]. In the basic implementation presented here [**107**], the algorithm is adapted to solve the SLO MINLP problem

$$\begin{aligned} \min_{X \in \Omega} \quad & f(X) \\ \text{subject to} \quad & c(X) \leq 0, \end{aligned}$$

where $X = (X_r, X_d) \in \Omega \subset \mathbb{R}^{n_r} \times \mathbb{Z}^{n_d}$. The optimization problem can be shortly represented as the triple $P = (f, c, \Omega)$. The box constraints defines the domain Ω while the inequality constraints c are treated independently. The BB method is based on the partition of the search space into N disconnected subsets

$$\Omega = \bigcup_{j=1}^{N} \Omega_j, \qquad \Omega_j \cap \Omega_i = \emptyset, \ \ \forall i \neq j.$$

The optimization subproblems $P_i = (f, c, \Omega_i)$ which mean

$$\begin{aligned} \min_{X \in \Omega_i} \quad & f(X) \\ \text{subject to} \quad & c(X) \leq 0 \end{aligned}$$

are such that the solution of the original problem (f, c, Ω) is solution of at least one of the subproblems (f, c, Ω_j)

$$S(f, c, \Omega) = \min\{S(f, c, \Omega_1), ..., S(f, c, \Omega_N)\}$$

where S indicates the solution of the corresponding optimization problem. The procedure of converging to the solution of the original problem solving one of the deriving subproblems generates a decision tree (Figure 3.1) where *selection* and *branching rules* determine which subproblem is solved next and with respect to which optimization variable the branching of the domain along the dimension corresponding to it should be performed.

To apply the procedure to MINLP problems it is necessary to introduce first the concept of continuous relaxation. The MINLP is said to be relaxed if the

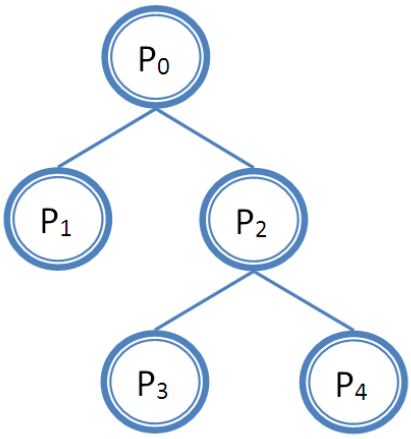


FIGURE 3.1. Branch and Bound decision tree.

discrete variables are treated as continuous variables

$$\begin{aligned} \min_{X\in\Omega} \quad & \widetilde{f}(X) \\ \text{subject to} \quad & \widetilde{c}(X) \leq 0 \end{aligned}$$

where $\Omega \subset \mathbb{R}^n$ with $n = n_r + n_d$. This way the model functions are evaluated in fractional values in correspondence of the discrete variables. The solution of the relaxed problem will be a lower bound for the solution of the original one,

$$S(\widetilde{f}, \widetilde{c}, \Omega) \leq S(f, c, \Omega).$$

The *active set* is the collection of subproblems (nodes of the tree) that have not been discarded but need to be investigated deeper.
The algorithm starts solving the relaxed optimization problem on the overall search domain. If an integer solution is found then the solution of the NLP problem is also solution of the MINLP, the node is not added to the active set and the algorithm terminates. Otherwise, the root problem is added as a pending problem in the active set and the cycle begins. The root node is picked up and removed from the active set. One of the discrete variables that assume fractional values in the optimal solution of the relaxed problem is selected and the search space is partitioned along its dimension. The generated two NLP subproblems are solved in parallel. The corresponding solutions and objective values are stored in the decision tree as lower bound of the MINLP problem. If an integer solution is found the upper bound of the objective value is updated with the optimal value of the subproblem and the index in the decision tree of the solution is stored in the memory as possible final optimal solution. If no integer solution is found, the subproblem is added to the active set for further branching. The process repeats until the active set is empty. When the process terminates, the solution in the decision tree corresponding to the best integer solution found is the optimal solution of the MINLP problem and lower and upper bound of the optimal value will be

identical.
The efficiency of a branch-and-bound algorithm is related to the branching and selection rules used: selection of the subproblems in the active set and branching along one of the integer variables that assumes fractional values. Several techniques have been developed during the years [**108, 109**], for example selection rules that exploit the information on the lower bounds for selecting the node with the lowest one or selection of the last added node in the decision tree. The identification of the optimization variables for performing the branching can be selected for example as the variable with the highest fractional part or the most sensitive variable of the objective function. Being x_k the discrete variable selected for the branching and x_k^* the fractional value assumed in the optimum of the subproblem, the search space is partitioned as $\Omega = \Omega_1 \cup \Omega_2$ where

$$\Omega_1 = \{X \in \Omega \mid x_k \leq \lfloor x_k^* \rfloor\}, \quad \Omega_2 = \{X \in \Omega \mid x_k \geq \lfloor x_k^* \rfloor + 1\}.$$

The algorithm can be summarized as follows

Algorithm 3.4.2. *BBWORHP*

(1) *Initialize the upper bound of the objective value* $f_{\max} = \infty$*, and the active set* $\mathcal{A} = \emptyset$*.*
(2) *Solve the continuous relaxation of the MINLP problem* $P_0 = (\widetilde{f}, \widetilde{c}, \Omega)$*.*
(3) *If* $S(\widetilde{f}, \widetilde{c}, \Omega)$ *is integer, terminate. Otherwise* $\mathcal{A} \leftarrowtail P_0$
   - *Select a subproblem* P_i *from the active set* $\mathcal{A}$*. The subproblem is removed from* $\mathcal{A}$*, and its solution is*
   $$S = S(\widetilde{f}, \widetilde{c}, \Omega_i).$$
   - *Branch the search space along the dimension corresponding to a non integer value of its optimal solution*
   $$\Omega_i = \Omega_{i_1} \cup \Omega_{i_2}$$
   - *Solve the two subproblems*
   $$S_1 = S(\widetilde{f}, \widetilde{c}, \Omega_{i_1}), \quad S_2 = S(\widetilde{f}, \widetilde{c}, \Omega_{i_2}).$$
   The problems are added to the decision tree storing the respective optimal values as lower bounds of the optimal value of the original problem.
   - *If* S_j *is integer with* $j = 1, 2$ *update of the optimal value upper bound*
   $$if(f_{S_j} < f_{\max}) \quad f_{\max} = f_{S_j},$$
   otherwise $\mathcal{A} \leftarrowtail P_j$*.*
   - *Iterate until* $\mathcal{A} = \emptyset$*.*
(4) *If* $f_{\max} \neq \infty$ *this is the minimum value attained by all the integer solutions found along the decision tree.*

Some further implementation of analogous algorithms can be found in [**110, 111, 112**]. The BB algorithm applied to the solution of MINLP can guarantee exact convergence to the global optimal solution of convex problems. The effective application of the algorithm is therefore strictly related to the provided first guess. If the optimization process starts in the region of attraction of the global optimum, the BBWORHP algorithm converges to the global optimal solution. On the other hand if the given initial guess is far from the optimum, then only local convergence can be guaranteed.
Moreover, the applicability of the algorithm is connected to the relaxation of the original MINLP problem. An optimization problem can be relaxed, if the functions involved in the model definition can be evaluated in fractional values of the discrete variables. This is not the case for example for *categorical* variables. Categorical variables are discrete variables that have just a qualitative meaning. Their integer values are just identification numbers that represent a particular quality, as for example the type of material of a particular component. If categorical variables are involved in the optimization process to apply the presented algorithm, it is necessary to act on the model itself as presented in [**113**].
Every categorical variable x, with N admissible values $x \in \{x_{c,1}, ..., x_{c,N}\}$ is substituted by a binary vector of dimension N, $b \in [0,1]^N$. Every function $h(x)$ depending on the categorical variables is substituted in the model by the function

$$h_c(b) := \sum_{i=1}^{N} b_i h(x_{c,i}).$$

The problem is now relaxable with respect to the binary variables b_i and the additional constraint $\sum_{i=1}^{N} b_i = 1$ needs to be included in the optimization problem to ensure that just one quality-value is selected for the x variable when converging to the optimal solution. This procedure has the major drawback of increasing the size of the discrete variables set and the number of constraints of the original problem, but renders the BBWORHP algorithm applicable to the complete set of MINLP problems.

3.4.2. Global Multi-Objective Optimization Techniques. For the multiple-objective case an heuristic stochastic strategy of the family of the evolutionary algorithms and an exact deterministic of the group of the derivative free optimization techniques have been selected and compared on a set of analytic and applied test problems. The two strategies are presented here in more detail.

3.4.2.1. *Hybrid Global Optimization (HGO).* It is a collaborative hybridization of three evolutionary algorithms. There is no evolutionary strategy that performs well for every kind of problems. Usually an elevate performance over one class of problems is paid in under-performance over another one (see *No Free Lunch Theorem* [**114**]). Here it comes the need of hybridization techniques, able to outperform the single optimization strategy on a wider

range of applications.
In collaborative or cooperative hybridization the algorithms involved act on mutually fine tuning the performance of the other with a continuous exchange of information during the optimization process. Different attempts have been done during the years combining genetic algorithms and swarm intelligence [**115, 116**] or genetic algorithms assisted by ant colony optimization algorithms [**117**] showing in all cases that the proposed hybridization performs better than the pure evolutionary strategy over complex nonlinear optimization problems, both in terms of speed of convergence and in the exploration capabilities of locating the global optimum.
The presented hybridization techniques implement the cooperative collaboration of three multi-objective evolutionary strategies: Double Grid Multi Objective Particle Swarm Optimization (DGMOPSO) [**118**], Non Dominated Sorting Genetic Algorithm V.2 (NSGA2) [**93**], Multi Objective Ant Colony Optimization for continuous domains (MOACOr) [**101**]. They have been selected on the base of their efficiency, in terms of function evaluations required for convergence and robustness on a set of benchmark problems analyzed in [**118**]. In particular NSGA2 presented the most robust behavior on various optimization problems confirming the high number of applications and research studies performed during the years, its major drawback is the low computational efficiency; MOACOr among the three algorithms is the most efficient one, but failed to converge on the most challenging problems; DGMOPSO is in between the two strategies, it shows average good results in terms of both robustness and efficiency but failed in converging to the global front for the binary test problem, outperformed in this case by the NSGA2 algorithm.
The strength of the three algorithms can be exploited with a parallel execution of them on different set of solutions and recombinations of the best offspring obtained in an iterative process with a continuous swap of information.
The initial solution set, randomly initialized, is equally divided between the three strategies. The three algorithms generate the derived set of solutions for a predefined number of sub-iterations, applying their own operators. The new generation is recombined, eventually with the solution stored in the external archive, sorted according to the non dominated sorting criteria and crowding distance operator of the NSGA2 algorithm (see [**89**] for details) to detect the Pareto optima and ensure a good spread over the front. The current optima are stored in an external archive and the contribution of each algorithm to the new solutions is weighted with hybridization coefficients that are used to distribute the new solution set among the three strategies in such way, that the optimization process steers toward the strategy that performed better in the previous iteration. The process iterates, starting from the best solutions found given to all the algorithms, until a stopping condition is encountered. To avoid premature exclusion of one of the algorithms a fixed threshold is set to guarantee a minimum amount of solutions. The corresponding percentage

of the entire population is ensured, to each algorithm at each iteration, in a way that if in a particular iteration the strategy under-performed is not kicked out from the optimization process. Other expedients have been implemented, such as the recombination of the dominated solutions if the non dominated solutions doesn't fill the entire archive. Instead of generating randomly the rest of the solutions needed to restart each algorithm, the best among the recombined set of dominated solutions is used. The algorithm can be summarized as follows

ALGORITHM 3.4.3. *HGO*

(1) *Initialization: random initialization of the initial population* P_0 *and equal partitioning among the different evolutionary algorithms*

$$P_0 = P_1 \cup P_2 \cup P_3, \quad P_i \cap P_j = \emptyset \ \ \forall i, j = 1, ..., 3 \ \ i \neq j.$$

Initialization of the external archive $\mathcal{A} = \emptyset$.

(2) *Iterative process:*

- *Evolution of the* P_j *solutions subset following the j-th strategy for all j=1,...,3 and generating the derived* R_j *solutions set of non dominated solutions and* D_j *the dominated one*

$$P_j \mapsto (R_j, D_j).$$

- *Recombination of the non dominated solutions with the solutions contained in the archive*

$$R = \mathcal{A} \cup R_1 \cup R_2 \cup R_3$$

and the dominated one

$$D = D_1 \cup D_2 \cup D_3.$$

- *Fast non dominating sorting and crowding distance assignment operator for sorting the solutions of* R *and* D *according to the non-dominance and uniform distribution criteria.*
- *Store the current non-dominated solutions in the archive* $\mathcal{A} \leftarrow\!\shortmid R^*$
- *Update of the hybridization coefficients* h_1 *and* h_2 *to weight the contributions of each algorithm to the front* R^*
- *Distribute the solution set* $\bar{R} = R^* \cup R \cup D$, *with* $|\bar{R}| = n$, *where* n *is the number of solutions of the entire population, obtained as the union, in order until the set if filled, of the best non dominated solutions* R^*, *the dominated one belonging to the frontiers of the different algorithms* R *and the dominated solutions* D *of each strategy*

$$P_1 = h_1 \bar{R}, \quad P_2 = h_2 \bar{R}, \quad P_3 = (1 - h_1 - h_2) \bar{R}$$

(3) *Termination: the process terminates when the maximum number of iterations is reached and the solutions stored in* $\mathcal{A}$ *are the approximation of the global Pareto frontier.*

The hybridization technique presented has been designed flexible enough to include just two of the selected strategies. Each algorithm has been re-adapted to allow archive dimensions greater than the population size, and the original implementation of the MOACOr algorithm has been enhanced including the mutation operator of NSGA2 to avoid premature convergence to a single solution in the front.
The algorithm terminates either if the maximum number of iteration is reached or if the Mutual Domination Ratio (MDR) or Consolidation Ratio (CR) operator falls below a predefined threshold. In [**119**] a survey and a comparison of the most recent metrics developed for estimating the convergence properties of multi-objective optimization strategies based on the notion of Pareto-dominance are given. The main idea of all the developed measures is the identification of a Progress Indicator (PI) and the monitoring of its convergence: when the improvements of the indicator are below a predefined threshold, the optimization algorithm is terminated to avoid waste of computational time. An online stopping criterion can formally be divided in the elements:

$$OSC := \{S, \Pi(\cdot), \Upsilon(\cdot), \phi(\cdot)\}$$

where S is the set of all the data elements involved in the optimization process required for the computation of the criteria [**119**]. $\Pi(\cdot)$ is the function for the Progress Indicator computation, $\Upsilon(\cdot)$ is the evidence gathering process, usually an aggregate function of the computed $\Pi(\cdot)$ with the scope of increasing the robustness of the process. The decision function $\phi(\cdot)$ determines where the optimization algorithm should stop because the criteria felt below a predefined threshold or continue with the approximation of the Pareto Front.
The selection of a useful metric for the given MDO problem is based on their efficiency in terms of no additional function evaluation for the assessment of the progress integrator. No knowledge about the analytic front is required and it is a generic strategy applicable to every evolutionary algorithm and it is not constrained to particular algorithm operators. Based on these considerations the best methodologies selected are the MGBM criterion [**120**] and the non-dominance based convergence metric [**121**]. The first one combine the MDR as PI with a simplified Kalman filter. The MDR indicator is based on the comparison of the solutions of the current Pareto front and the ones of the front at the previous iteration, checking how many individuals of the new front are dominating the one of the previous one and viceversa

$$I_{\mathrm{mdr}}(\mathcal{A}_k, \mathcal{A}_{k-1}) = \frac{|\Gamma(\mathcal{A}_{k-1}, \mathcal{A}_k)|}{|\mathcal{A}_{k-1}|} - \frac{|\Gamma(\mathcal{A}_k, \mathcal{A}_{k-1})|}{|\mathcal{A}_k|},$$

where $\mathcal{A}_k$ is the Pareto front at the k-th iteration stored in the archive and $\Gamma(A, B)$, with A and B generic set of solutions is the function that returns the set of element of A that are dominated by at least one element of B

$$\Gamma(A, B) = \{x \in A \mid \exists y \in B \text{ with } y \preceq x\}$$

and $|A|$ is the number of elements of the set A. It is suitable for solving large scale problems since no additional model evaluations are needed and relying only on the already computed solutions of the Pareto front.
The second metric, the non-dominance based convergence metric, has already been successfully applied to a crashworthiness MDO problem. The Progress Indicator is the consolidation ratio defined as

$$I_{\mathrm{cr}} = \frac{|\mathcal{S}|}{|\mathcal{A}_k|}, \quad \mathcal{S} = \{a_{k-\tau} \in \mathcal{A}_{k-\tau} \mid \exists a_k \in \mathcal{A}_k \text{ with } a_{k-\tau} = a_k\}$$

intended as the portion of solutions in the current Pareto front that are maintained with respect to the front obtained τ iterations before. An utility function that reduces the noise of the CR computation is introduced to prevent premature stopping of the algorithm, for trapping in local minima. It is a scaling of the difference between the CR operators values at the current iteration k and the one at the $k - \tau$ iteration. No consecutive iterations are taken to increase the robustness of the process.
The main weakness of evolutionary algorithms is their efficiency, measured in number of function evaluation, when compared to traditional optimization methods. One model evaluation needs to be performed for every member of the solution set at every iteration of the algorithm. If the search space is large, a higher number of solutions are involved in the search process and the computational load increases significantly. A possibility to overcome this difficulty is the parallelization of the algorithm. The nature of the population-based algorithms is well suited for a parallel architecture: the evaluation of the model in the different solutions can be executed in parallel on different processors. An OpenMP [**122**] parallelization of all the evolutionary strategies, including the hybrid algorithm, is implemented in the developed software. OpenMP is a programming interface that supports multiplatform shared memory multiprocessing programming. Shared memory is a memory than can be accessed simultaneously by multiple process to communicate and share informations between them. The series of instructions is divided among different threads executed in parallel, with one thread holding the role of the master thread. The model evaluations are performed by each thread on a given subset of the whole set of solutions. The access and the update of the necessary data passes through the shared memory. After the execution of the parallelized code the threads gather back together into the master thread, which recombines, sorts and applies the algorithm specific operators. OpenMP is easily implementable on modern multicore laptops. For the research purpose of developing a tool which is able to run on a single machine, it has been selected as the parallelization strategy among the well known Message Passing Interface (MPI) parallelization for distributed memory machines and Graphich Programming Unit (GPU). The time required for the execution of a parallel process is

$$T' \simeq \frac{T}{n}$$

where n is the number of cores available and T the time employed for the sequential execution.

3.4.2.2. *Mesh Adaptive Direct Search (MADS).* The MADS algorithm is an exact deterministic technique belonging to the class of sequential approximation and derivative free optimization strategies. No evaluations of the objective and constraints derivatives are required during the optimization process, so the algorithm can be effectively applied to black box optimization problems where the computation of the derivatives, even by finite difference is impractical. This is the case for example of non continuous functions or discrete variables. The MADS approach presented here is for single objective optimization, the extension to multiple objectives is presented below with the name BIMADS.
A hierarchy of convergence results is given in [**123**] with different assumptions on the problem smoothness. In particular at the two extremes of the scale there are the following results

STATEMENT 3.4.1. If the iterates produced by the algorithm are bounded, then there is a $X^* \in \Omega$ which is the limit of mesh local optimizers on meshes that get infinitely fine.

STATEMENT 3.4.2. Let the iterates produced by the algorithm are bounded, the objective f is strictly differentiable near the limit X^* and the constraint qualification, that the tangent cone $T_\Omega(X^*)$ to the feasible region Ω at $X^* \in \Omega$ is non-empty and full-dimensional. Then the directional derivatives satisfy

$$\nabla f(X^*) \cdot d \geq 0 \quad \text{for all } d \in T_\Omega(X^*).$$

The first result applies on very mild assumptions, the request of having a bounded sequence of iterates holds, if the box constraints don't have unbounded limits. Then it's been proved that the method converges locally to a solution in the feasible domain. In the second result, the additional requirement is the differentiability of the objective function; under these hypothesis it holds the first order optimality KKT condition that there are no feasible strict descent directions from the point X^*.
The MADS algorithm is an evolution of the Generalized Pattern Search (GPS) method [**124**], with improvements in the search directions strategy. The basic idea of both methods is to generate successive trial points on a tower of underlying meshes that adapt their finesse approaching the optimal point. At each iteration a finite number of trial points are spawned, the evaluation of the model on the set of points is performed first for the feasibility evaluation. If a point is infeasible, it is filtered and no computational power is lost in the evaluation of the objective. Otherwise, if the trial point lies in the feasible region, its objective value is compared with the one of the current incumbent solution (the best solution obtained at the previous step). At the k-th

iteration the trial points lie on a mesh defined as

$$M_k = \bigcup_{X \in V_k} \{X + \triangle_k^m Dz \,|\, z \in \mathbb{N}^{n_D}\},$$

where $\triangle_k^m \in \mathbb{R}^+$ is the *mesh size* parameter at iteration k, that defines the finesse of the mesh, $D \in \mathrm{Mat}(n, n_D)$ is the *mesh directions* matrix that represents a set of n_D directions in $\mathbb{R}^n$ with n the problem dimension and V_k is the current set of trial points.
Each iteration is divided into two steps. The first is called *search* step, the points lying on the mesh are examined for finding an improved solution, unfeasible points or points that are not reducing the objective value are discarded. If an improved mesh point is detected a new iteration starts with the new solution as the incumbent solution for generating the new set of trial points. The mesh size is updated. If the search step fails to generate an improved mesh point, the *poll* step will be invoked. This is the main difference between MADS and GPS. A new pool size parameter Δ_k^p is introduced to allow a wider exploration of the feasible space. A new set of trial points are generated on a so called *frame* defined as

$$P_k = \{X_k + \Delta_k^m d \mid d \in D_k\} \subset M_k,$$

where D_k is a positive spanning. The distance from the incumbent solution X_k and the frame point $X_k + \Delta_k^m d \in P_k$ is bounded by a constant multiplied by the pool size parameter

$$\Delta_k^m \|d\|_2 \leq \Delta_k^p \max\{\|d'\|_2 \mid d' \in D\}.$$

The mesh size parameter is updated according to

$$\Delta_{k+1}^m = \tau^{w_k} \Delta_k^m,$$

where $\tau \in \mathbb{Z}$, $\tau > 1$ and $w_k \in \mathbb{Z}$, such that it takes non-negative values if the k-th iteration was successful, negative otherwise. In such way the mesh size parameter decreases after a failing step and can increase after a successful one. The pool size parameter is updated according to the pool strategy following the rule that the mesh size parameter is smaller than the pool one, and reduces faster after a failure (see Figure 3.2 as example).
The algorithm can be summarized in the following steps:

Algorithm 3.4.4. *MADS*

(1) *Initialization:* $X_0 \in \Omega$ *is the incumbent solution and* $f_\Omega \leftarrow\!\shortmid f(X_0)$ *the best objective value attained. The mesh and poll parameters* $\Delta_0^m \leq \Delta_0^p$ *are initialized with the requirements given above.*
(2) *Iteration* $k = 0, ..., N_{\max}$*: Perform search and poll steps until an improved mesh point* X_{k+1} *is found on the mesh* M_k*. Whenever an improved point is found* $f_\Omega \leftarrow\!\shortmid f(X_{k+1})$ *is the current approximation of the minimum value*
   - *Search: evaluate* f *on a finite subset of trial points on the mesh* M_k

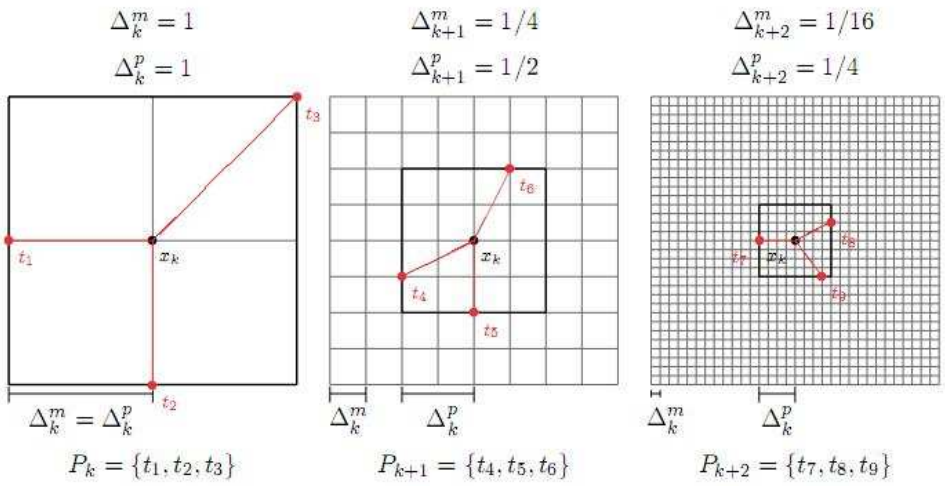


FIGURE 3.2. Example of different mesh configurations with $n = 2$. Thin lines represent the mesh of size Δ_k^m, and thick lines the points at distance Δ_k^p from X_k in norm L1. The mesh is reduced between the situations in order to show that Δ_k^m is reduced faster than Δ_k^p. As the poll trial points lie at the intersection of the thick and thin lines, the number of possible locations grows larger and larger.

- *Pool: evaluate f on the frame P_k*
- *Update: update Δ_k^m and Δ_k^p*

(3) *Termination: the process terminates when the maximum number of black box evaluations is reached. f_Ω is the best objective value achieved.*

The presented approach to address constraints is called *extreme barrier*: a step is successful only if a new best feasible solution is found. Unfeasible points are rejected. Another way of handling constraints is through *the filter and the progressive barrier techniques*, where unfeasible points are not discarded but the magnitude of constraints violation is evaluated and included in the final objective value (see [**125**] for details).
Also different search and pool strategies have been studied. The most efficient search strategy is the *surrogate* approach. A surrogate is an inexpensive function, that the user can employ for efficiently evaluating the problem functions, and to explore extensively the current mesh for points that the surrogate predicts to reduce the number of function evaluations. For example it is possible to apply a local method to the surrogate problem, generate several good local optimizers for that problem and use the expensive true objective and constraints functions at those points to decide whether the search has been successful or not. The values of the true problem computed are used to improve and recalibrate the surrogates.
The different *pooling strategies* are distinguished by the different search directions used (see Figure 3.3). MADS allows a larger set of pool directions

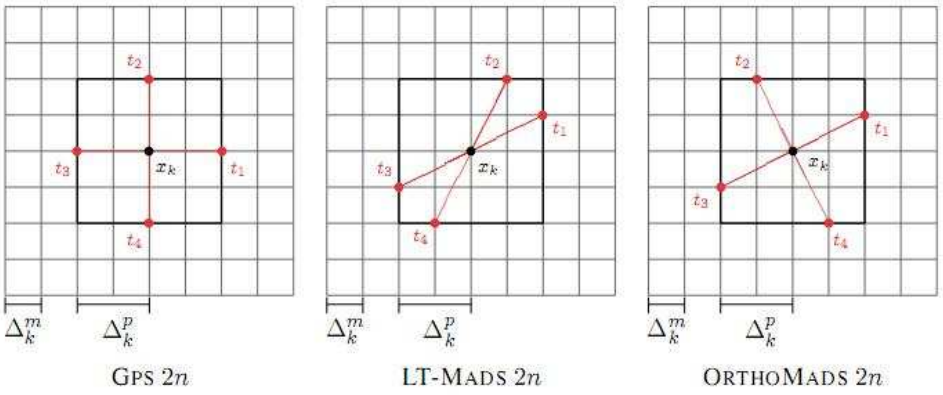


FIGURE 3.3. Illustration of the differences between the types of directions in the case $n = 2$. In the three cases, $\Delta_k^p = 2\Delta_k^m$, and $2n$ directions are considered. GPS directions are orthogonal but correspond always to the basis directions. LT-MADS and ORTHOMADS have greater flexibility: trial points can be anywhere at the intersection of thin and thick lines. Moreover, ORTHOMADS directions are orthogonal.

compared to GPS. In fact as k goes to infinity the union of the normalized pool directions over all k becomes dense in the unit sphere. If some partial derivatives are known MADS can exploit this information to reduce the number of function evaluations in each pool step directing the search in the direction of steepest descent. For integer variables, directions are computed the same way except that round-offs are performed. Special directions are used for binary variables.

The MADS algorithm has not been implemented from scratch. The SLO layer includes the Nonsmooth Optimization by Mesh Adaptive Direct Search (NOMAD) tool [**126**] that implements the presented methodology. NOMAD is an open source, multiplatform C++ library, under development since 2000. Its first version implemented the GPS algorithm. MADS was developed in 2006 and integrated into NOMAD since then.

Only bi-objective optimization problem can be solved with NOMAD. The BIMADS algorithm is launched, instead that the single objective MADS, that consists in a run of a series of single-objective MADS runs to build up a list of not dominated solutions and approximate the Pareto front. For more details see [**127**].

NOMAD is at its best for black-box problems with less than 50 variables. For applicative test problems see [**128**]. In particular an analyzed test case leading to promising results is a small MDO problem for preliminary structure, aerodynamic and propulsion aircraft design can be found in [**129**].

For larger problems, the Parallel Space Decomposition of the Mesh Adaptive Direct Search (PSDMADS) [**130**] algorithm, in which the optimization processes solve successive problems over subsets of variables, should be considered. PSDMADS has been tested for problems with up to 500 variables

illustrating some advantages and some limitations [**130**]. PSDMADS is not yet integrated in the NOMAD package, but it will be in a future release.

3.4.3. Local Optimization Techniques. Among all the gradient based optimization strategies the combined SQP-Interior Point (IP) method implemented in the WORHP[1] library has been selected to efficiently solve NLP problems. The methodology, since based on the computation of derivatives, is not applicable to optimization problems involving discrete variables. The local optimization is used in the MDO of launchers just for the local refinement of global solutions, fixing the value of the discrete variables or for efficiently solving the trajectory optimization subproblems that include just continuous control variables.
WORHP is a sparse large-scale NLP solver aiming to solve optimization problems with more than billions of variables and constraints. It has been developed by the joint work of teams from the University of Bremen and University of Würzburg (see [**131**]).
The general idea of SQP methods was introduced by Han in 1977 [**132**] and earlier by Wilson in 1963. Since then they belong to the most frequently used algorithms for the solution of practical optimization problems due to their robustness and their good convergence properties.
The original NLP problem is locally approximated around the X_k iterate by a quadratic problem of the form

$$\begin{aligned} \min_{d\in\mathbb{R}^{n_x}} \quad & \tfrac{1}{2} d^T \nabla^2_{xx} \mathcal{L}(X_k, \mu_k) d + \nabla f(X_k)^T d \\ \text{subject to} \quad & g_i(X_k) + \nabla g_i(X_k)^T d \leq 0, \quad i = 1, ..., m, \end{aligned}$$

which solution $d \in \mathbb{R}^{n_x}$ is the search direction used to define the next iterate $X_{k+1} = X_k + \alpha d$. The solution of the quadratic problem is approximated with a Primal-Dual Interior Point method [**134**]. The step size $\alpha \in \mathbb{R}$ is instead chosen according to a measure of quality provided by a *merit function*, or penalty function that defines a balance between optimality and feasibility (penalty parameter $\eta \in \mathbb{R}^m$), here defined as

$$L_1(X, \eta) = f(X) + \sum_{i=1}^{m} \eta_i \max(0, g_i(X)).$$

In order to improve the robustness of the algorithm with respect to the accuracy of the merit function, a series of recovery strategies have been implemented in the code to avoid premature stopping of the optimization process. To determine the step size, an auxiliary function is defined

$$\phi(\alpha) = L_1(X_k + \alpha d_k, \eta_k).$$

Line search is applied to provide sufficient decrease of the merit function evaluating it in a decreasing series of stepsizes $\{\alpha_i\}_i$ and selecting the first

[1]We Optimize Really Huge Problems (WORHP)

one, that satisfies the Armijo condition

$$\phi(\alpha_i) \leq \phi(0) + \sigma\phi'(0).$$

An alternative to the use of a merit function is the *filter*, a two-dimensional set of points that considers the objective function value and constraint violation separately, and accepts trial points, if they yield an improvement in either direction [**135**].
The determination of derivatives is a crucial element in nonlinear optimization. Basically, first derivatives as the gradient of the objective function or the Jacobian matrix of the constraints are necessary in order to find the descent direction d_k and the second derivatives (Hessian of the Lagrangian) are used for local quadratic approximations at the iterates and to improve the descent direction. If the derivatives are not given by the user, WORHP provides several alternatives, e.g. a method based on group strategies for finite differences or new sparse Broyden-Fletcher-Goldfarb-Shanno (BFGS) update techniques to obtain an approximation of the Hessian. The algorithm can be summarized in the following steps:

Algorithm 3.4.5. *WORHP*

(1) *Initialization: initialize X_0 with the given initial guess.*
(2) *Iteration $k = 0, ..., N_{\max}$:*
   - *Approximate the nonlinear problem by a quadratic subproblem in X_k and use its solution d_k as the search direction.*
   - *Determine a step size α_k by applying a line search method to a merit function or using filtering techniques*
   - *Update the iterate $X_{k+1} = X_k + \alpha_k d$ and increment k*
   - *Iterate until termination criterion is met or maximum number of iteration is reached*
(3) *Termination: the algorithm stops whenever the KKT optimality conditions are satisfied for the level of accuracy defined by the user.*

The WORHP architecture is based on a reverse communication that offers unique flexibility and control over the optimization process: the intervention between the different calls to the model, gradient and Hessian evaluation, optimality condition check and creation and resolution of the QP-subproblem is permitted at each step of the optimization process.
The robustness of WORHP was proved by the CUTEr test set, which consists of 920 sparse large-scale and small dense problems, where WORHP solves 99.5% of all test cases.

3.4.4. Reverse Communication. All the presented global and local optimization techniques for NLP/MINLP problems are embedded in a reverse communication architecture. The SLO MDO problem doesn't want to be an automatic way of operating but the intervention of the user in the optimization process to steer toward best iterates and faster convergence to the global optima must be considered when designing the SLO architecture.

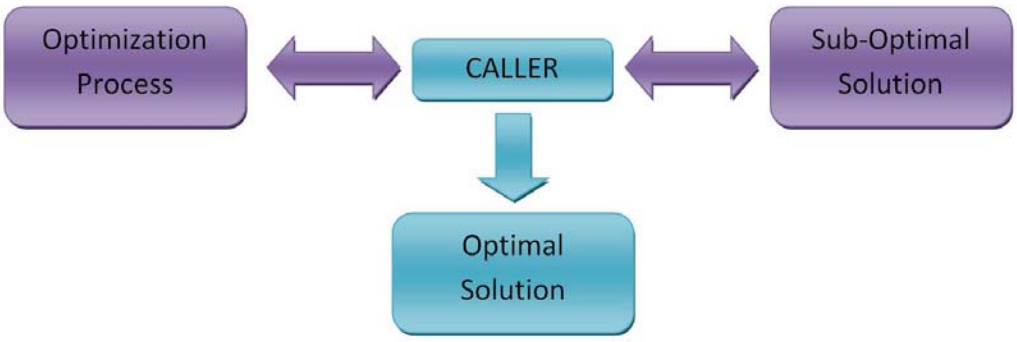


FIGURE 3.4. SVAGO Reverse Communication architecture

A predefined number of iterations for the reverse communication loop is set by the user at the starting point of the optimization process. Whichever algorithm is chosen at every iteration, the control goes back to the master loop that determines if the process needs to be hybridized for external investigation or proceeds to the next step. When the control goes back to the user, a default waiting time frame of 1 minute is activated during which time the user can choose between hybridizing the process and have access to the current incumbent solution or frontier, or continuing to the next level. If the process is stopped, the algorithm parameters as well as the incumbent solution can be modified and the iteration can be restarted from the current setting reloading the latest solutions and performing the remaining steps (Figure 3.4).
A schematic representation of the complete SVAGO SLO layer is given in Figure 3.5. The distinction has been made between the class of algorithms they belong to: deterministic or stochastic, the number of objectives: in orange the multi-objective strategies and in red the single objective ones, the set of optimization variables they can deal with: X_R is the set of continuous variables, X_I are the integer variables and X_D the discrete variables including both integer and categorical variables. For the evolutionary strategies the possibility of activating the code parallelization using OpenMP paradigm and the automatic stopping criterion based on the information coming from the achieved frontier (MGBM and CR) are also outlined in the graph. These are all enhancements included in the developed tool that the user can choose to activate. Since the deterministic BBWORHP technique makes use of a continuous relaxation of the MINLP problem, the implemented reformulation of the problem functions, as explained in Sub-Section 3.4.1.2, is highlighted by the change from (X_R, X_I) to (X_R, X_D) as the set of variables the algorithm is able to deal with.

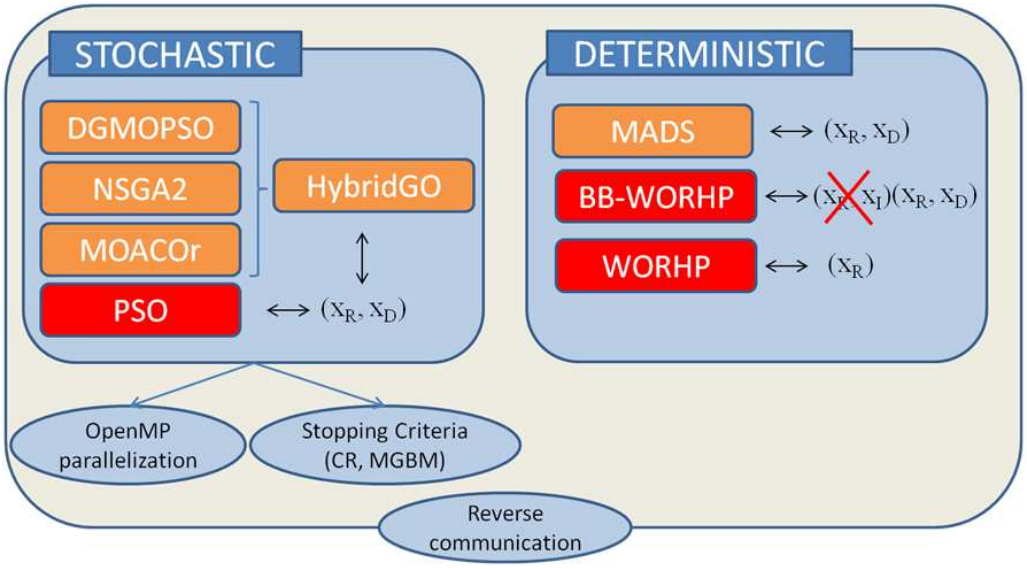


FIGURE 3.5. SVAGO SLO layer

CHAPTER 4

Optimal Control Problem

Contents

4.1.	**Optimal Control**	**89**
4.2.	**The Ascent Trajectory Optimization Problem**	**91**
4.2.1.	Dynamic Model	94
4.2.2.	Guidance Model	95
4.3.	**Multiple Shooting Techniques**	**97**
4.4.	**Semianalytic Methods**	**98**

The problem of optimizing the launcher trajectory is an optimal control problem. Optimal control theory has been formulated as an extension of calculus of variations and the minimum principle, developed in the 1950s by Pontryagin and one of his greatest achievements [**136**].

4.1. Optimal Control

An optimal control problem is intended as the process of finding the control laws for a given system that minimizes a cost functional subject to initial and final states as well as path constraints. Generally, a control function $u(t) \in \mathcal{C}_p^0([t_0, t_f]; \mathbb{R}^{n_c})$ has to be determined in order to influence the changes in the state function $x(t) \in \mathcal{C}_p^0([t_0, t_f]; \mathbb{R}^{n_s})$, where n_s are the number of states and n_c the dimension of the control space. $\mathcal{C}_p^j(I; \mathbb{R}^n)$ denotes the class of j times piecewise continuously differentiable functions from the interval $I \subset \mathbb{R}$ to $\mathbb{R}^n$. The system dynamics is defined by the Ordinary Differential Equation (ODE) system

$$\dot{x}(t) = f(x(t), u(t))^1, \quad t \in [t_0, t_f],$$

with $f : \mathbb{R}^{n_s} \times \mathbb{R}^{n_c} \to \mathbb{R}^{n_s}$ continuously differentiable with respect to x and u. The initial and final states can be set through the equation

$$\omega(x(t_0), x(t_f)) = 0$$

[1]For optimal control problems in astrodynamics it generally applies the non autonomous formulation (i.e with the explicit dependance over the time $f(x(t), u(t), t)$). The results reported here refer to the autonomous form because non autonomous problem can be transformed in autonomous one defining the time as additional state variable.

with $\omega : \mathbb{R}^{n_s} \times \mathbb{R}^{n_s} \to \mathbb{R}^{n_r}$ continuously differentiable. Constraints on the controls and states can be formulated as inequality conditions of the form

$$c(x(t), u(t)) \leq 0, \quad t \in [t_0, t_f]$$

with $c : \mathbb{R}^{n_s} \times \mathbb{R}^{n_c} \to \mathbb{R}^m$ continuously differentiable. The solution of an optimal control problem is the pair $(x^*(t), u^*(t))$ which minimize the objective

$$F[x, u] = \phi(x(t_f)) + \int_{t_0}^{t_f} f_0(x(t), u(t))\, dt$$

over all the admissible solutions $(x(t), u(t))$ with $\phi : \mathbb{R}^{n_s} \to \mathbb{R}$ and $f_0 : \mathbb{R}^{n_s} \times \mathbb{R}^{n_c} \to \mathbb{R}$ continuously differentiable.
Hence the standard optimal control problem is formulated as

$$\begin{array}{lrl} \min\limits_{u,x} & \phi(x(t_f)) + & \int_{t_0}^{t_f} f_0(x(t), u(t))\, dt \\ \text{subject to} & \dot{x}(t) & = f(x(t), u(t)), \\ & c(x(t), u(t)) & \leq 0, \quad t \in [t_0, t_f], \\ & \omega(x(t_0), x(t_f)) & = 0. \end{array}$$

The problem of finding the control law $u(t)$ is an infinite dimensional optimization problem, for which usually it is not possible to find the exact analytic solution. The numerical methods for the approximation of the optimal control profile are divided into indirect and direct methods.
Indirect methods were the mostly used in the early years of optimal control theory [**137**]. They convert the optimal control problem into a boundary value problem using the necessary condition of the Pontryagin minimum principle. For the unconstrained case, where $m = 0$, the Hamiltonian is defined as

$$H(x, u, \lambda_0, \lambda) = \lambda_0 f_0(x, u) + \lambda^T f(x, u)$$

with $\lambda_0 \in \mathbb{R}_0^+$ and the adjoints $\lambda \in \mathcal{C}_p^0([t_0, t_f]; \mathbb{R}^{n_s})$. The minimum principle then states as follow

THEOREM 4.1.1 (Pontryagin minimum principle). *Let $(x^*(t), u^*(t))$ be a solution of the unconstrained optimal control problem. Then there exist $\lambda_0 \in \mathbb{R}_0^+$, $\rho \in \mathbb{R}^{n_r}$, $\lambda \in \mathcal{C}_p^0([t_0, t_f]; \mathbb{R}^{n_s})$ not all vanishing, so that these equations holds:*

Minimum condition.

$$H(x^*(t), u^*(t), \lambda_0, \lambda(t)) = \min_{u(t) \in U} H(x^*(t), u(t), \lambda_0, \lambda(t)),$$

for almost all $t \in [t_0, t_f]$ and $U \subset \mathbb{R}^{n_c}$ defined by the box constraints on the control u.

Adjoint differential equations.

$$\dot{\lambda}(t) = -\nabla_x H(x^*(t), u^*(t), \lambda_0, \lambda(t)),$$

for almost all $t \in [t_0, t_f]$

Transversality conditions.

$$\begin{aligned} \lambda(t_0)^T &= -\nabla_{x(t_0)} \left(\rho \omega(x^*(t_0), x^*(t_f))\right), \\ \lambda(t_f)^T &= \nabla_{x(t_f)} \left(\lambda_0 \phi(x^*(t_f)) + \rho^T \omega(x^*(t_0), x^*(t_f))\right). \end{aligned}$$

For the proof of the theorem and for the results regarding the mixed constrained case refers to [**138**] and to [**139**] for state constraints. General results are yet to be proven [**140**].
The indirect methods have the disadvantage that solving the boundary value problem could be extremely difficult. Moreover, for the constrained case, the behavior of the controls along the boundaries of the domain needs to be known in advance. Otherwise all the possible combinations of the control structure need to be checked in order to apply the results.
In recent years, direct methods developed and established as the most effective methodology to solve optimal control problems. They are based on the discretization of the control and state functions, and approximation of the infinite dimensional optimal control problem by an NLP problem, see [**141**]. The continuous time axis $t \in [t_0, t_f]$ is replaced by a set of $r \in \mathbb{N}$ discrete grid points, $t \in \{t_0, t_1, ..., t_r\}$ with $t_r = t_f$. The control and state functions are approximated by interpolations on the set of grid points $\{u_0, u_1, ..., u_r\}$ and $\{x_0, x_1, ..., x_r\}$ where

$$u_i \approx u(t_i), \quad x_i \approx x(t_i).$$

Using a numerical method for the approximation of the integral in the objective function and for the resolution of the ODEs system, it is possible to transcribe the optimal control problem into an NLP problem of the form

$$\begin{aligned} \min_{\{u_i\},\{x_i\}} \quad & \phi(x_r) + \sum_{i=1}^{r-1} (t_{i+1} - t_i) f_0(x_i, u_i) \\ \text{subject to} \quad & x_{i+1} = x_i + (t_{i+1} - t_i) f(x_i, u_i), \quad i = 1, ..., r-1 \\ & c(x_i, u_i) \leq 0, \quad i = 1, ..., r, \\ & \omega(x_0, x_r) = 0, \end{aligned}$$

where the Euler method is used to approximate the integrals. The large and sparse NLP problem that follows can be efficiently solved using NLP solvers that are able to exploit their sparsity structure.
In the following sections the optimal control problem of optimizing the rocket trajectory is presented in its mathematical formulation and the approximation of its solution with direct methods is discussed.

4.2. The Ascent Trajectory Optimization Problem

The optimization of the rocket trajectory is an optimal control problem. The state variables are the position and the velocity coordinates of the launcher in space, the control variables are the attitude angles that define the direction of the thrust and the cost function is the maximization of the payload mass. The boundary constraints are the position at launch and the target orbit while the path constraints are structural constraints: dynamic pressure, axial acceleration and heat flux, which should not exceed a predefined maximum level.
In the two Subsections that follow, 4.2.1 and 4.2.2, the detailed formulation of the dynamics and guidance profiles implemented in the trajectory model

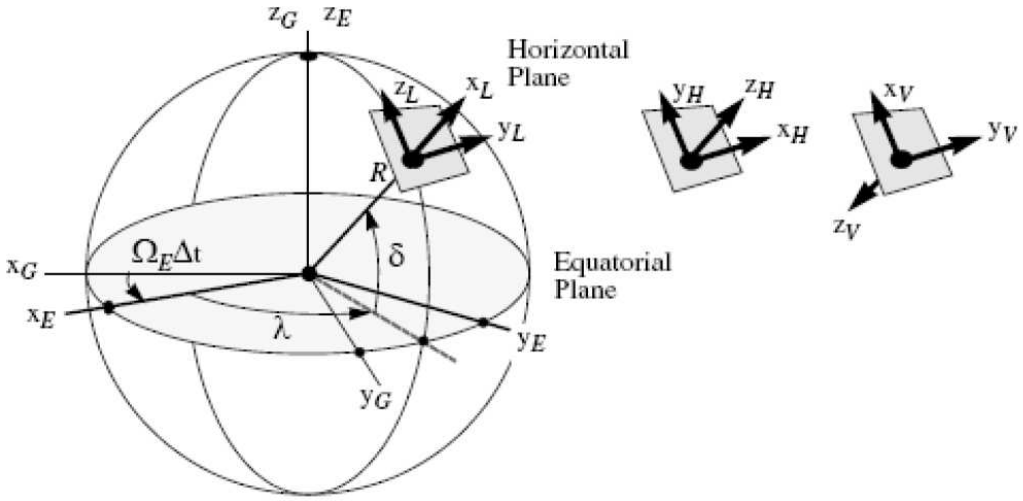


FIGURE 4.1. Earth Centered Inertial Frame and Local Coordinate System, [**118**].

are discussed. In order to fully understand the notation used, some reference frames need to be defined. Refer to Figure 4.1 for the first two definitions.

DEFINITION 4.2.1. *Earth Centered Inertial (ECI) Frame.* The origin of the coordinate system is the center of the Earth. The z_G axis extends through the North Pole. The x_G and y_G axes span the equatorial plane, where the x_G points into a direction fixed in space (the vernal equinox) and y_G is ortogonal to x_G and z_G.

The dynamic of a point in space in the ECI system is defined through a set of 6 Cartesian coordinates: 3 representing the components of the position vector and 3 for the velocity.

DEFINITION 4.2.2. *Local Coordinate System.* The system originates at the launcher position, defined by the radius r, longitude λ and latitude values δ. x_V and y_V span the local horizontal plane (the plane tangential the Earth's surface in the launcher position), with x_V pointing North and y_V East. The z_V axis points radially downward the plane along the local vertical, perpendicularly to the local horizontal plane.

The Relative Velocity System of Figure 4.2 is used for the definition of the launcher dynamics.

DEFINITION 4.2.3. The *Relative Velocity Frame* defines the velocity vector with respect to the local horizontal plane. The system originates at the launcher position, x_T is parallel to the direction of the velocity, y_T is perpendicular to x_T in the horizontal plane and z_T follows. The plane spanned by x_T and z_T is called local vertical plane.

A point moving in space is defined by 6 spherical coordinates enumerated in Table 4.1, 3 for the position and 3 for the velocity (2 angles and the vector

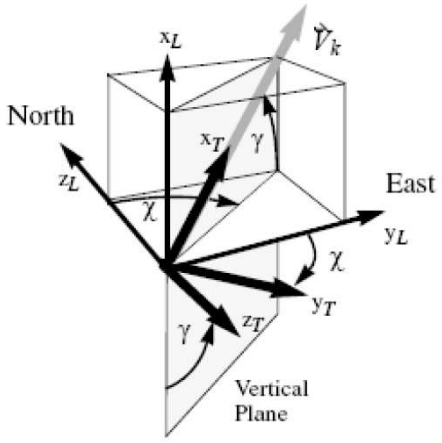


FIGURE 4.2. Relative Velocity Frame, [**118**].

r	Radial distance from the planet center
λ	Longitude, angle between the Greenwich meridian and the meridian of the current position, positive east of the reference meridian, $\lambda \in [0, 2\pi]$.
δ	Declination, angle between the equatorial plane and the current position, positive on the northern hemisphere, $\delta \in [-\pi/2, \pi/2]$.
V	Flight-path velocity, magnitude of the vehicle velocity relative o an Earth-fixed observer.
γ	Flight-path angle, angle between the flight-path velocity and the local horizontal plane, positive for ascent, $\gamma \in [-\pi/2, \pi/2]$.
φ	Flight-path heading, angle between the North and the flight-path velocity component projected onto the local horizontal plane, measured clockwise from the North. The value is undetermined above the poles and in vertical flight, $\varphi \in [0, 2\pi]$.

TABLE 4.1. Flight-path state variables definition

magnitude). To define the attitude of the launcher, that is the orientation of the rigid body with respect to a reference frame, three more angles and a new system rigidly attached to the vehicle's geometry needs to be defined. The additional variables are listed in Table 4.2 and displayed in Figure 4.3. In the current modeling no variation of the roll angle is considered, fixing its value to zero.

DEFINITION 4.2.4. *Body System.* The x_B and z_B span the vehicle's plane of symmetry, the x_B axis pointing forward, the z_B downward. The y_B axis is perpendicular to the vehicle's plane of symmetry pointing starboard.

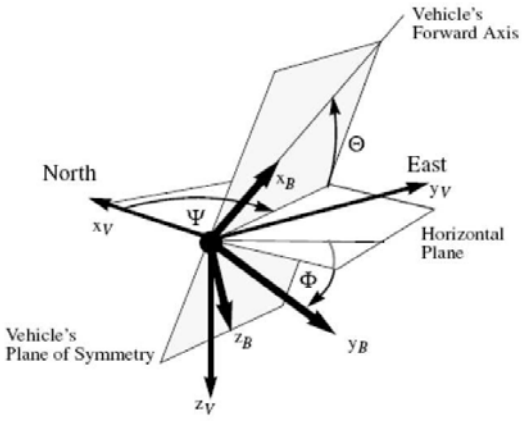


FIGURE 4.3. Body fixed system, [**118**].

ψ	Yaw. It is the angle between the vehicle's plane of symmetry and the x_v axis in the local coordinate system
θ	Pitch. It is the angle between the local horizontal plane and the vehicle's forward axis on his plane of symmetry. It is the sum of the flight path angle and the angle of attack (the angle between the relative velocity vector and the launcher axis)
ϕ	Roll. It is the angle between the perpendicular to the plane of symmetry and the local horizontal plane

TABLE 4.2. Attitude control angles

4.2.1. Dynamic Model. The 3 Degree of Freedom (DoF) model of the Equations of Motion (EoM) is given in the Relative Velocity Frame using spherical coordinates. The system is not inertial because it rotates with the Earth, so Coriolis and centrifuge accelerations must be taken into account:

$$
\begin{cases}
\dot{r} &= V\sin\gamma \\
\dot{\lambda} &= \dfrac{V\cos\gamma\sin\varphi}{r\cos\delta} \\
\dot{\delta} &= \dfrac{V\cos\gamma\cos\varphi}{r} \\
\dot{V} &= X+\omega_E^2 r\cos\delta(\cos\delta\sin\gamma-\sin\delta\cos\gamma\cos\varphi) \\
\dot{\gamma} &= -\dfrac{Z}{V}+\dfrac{V\cos\gamma}{r}+2\omega_E\cos\delta\sin\varphi+ \\
&\quad +\omega_E^2 r\cos\delta\dfrac{\cos\delta\cos\gamma-\sin\delta\sin\gamma\cos\varphi}{V} \\
\dot{\varphi} &= \dfrac{Y}{V\cos\gamma}+\dfrac{V\cos\gamma}{r}\tan\delta\sin\varphi+2\omega_E(\sin\delta-\cos\delta\tan\gamma\cos\varphi)+ \\
&\quad +\omega_E^2 r\cos\delta\dfrac{\sin\delta\sin\varphi}{V\cos\gamma}
\end{cases}
$$

Where ω_E is the angular velocity of the planet and X, Y, Z are the components of the total external acceleration, which is the sum of thrust, gravitational and aerodynamic accelerations (lift and drag) in the local reference frame. The accelerations defined in the own reference frames are

$$\begin{aligned} G_{\text{acc}} &= g(r) \cdot z_V, \\ T_{\text{acc}} &= \frac{T}{m(t)} \cdot x_B, \\ D_{\text{acc}} &= -\frac{\rho(r) V^2 A_{\text{ref}} C_D(M, \alpha)}{2m(t)} \cdot x_T, \\ L_{\text{acc}} &= -\frac{\rho(r) V^2 A_{\text{ref}} C_L(M, \alpha)}{2m(t)} \cdot x_T. \end{aligned}$$

By means of rotations, they can be transformed in the relative velocity frame. $g(r)$ is the gravity acceleration depending on the radial position, T is the thrust magnitude, the thrust is aligned with the x_B axis, m the total vehicle mass, $\rho(r)$ the air density depending on the altitude, V the velocity magnitude, the velocity is aligned with the x_T axis, A_{ref} the area of the vehicle surface where the aerodynamic forces applies, C_D and C_L the lift and drag coefficients, depending on the Mach number M and the angle of attack α.
The integration of the dynamics is done with two different approaches: first, with numerical methods, starting from the lift-off instant and stopping at the last stage burn out; and then with semianalytic methods: an analytic solutions is sought on smaller intervals by means of Taylor expansions and appropriate manipulations. These methodologies are explained respectively in Section 4.3 and Section 4.4.

4.2.2. Guidance Model. The guidance variables are the yaw angle for the horizontal guidance and the pitch angle for the vertical one. As stated above for simplicity, the roll angle is considered constant to zero. Optionally, also the thrust level can be controlled. In each phase a standard guidance law for each control angle is defined. The angle itself is then defined as the sum of the reference value and a deviation from it. The time interval of each phase is indicated as $[t_{F_1}, t_{F_2}]$ to indicate respectively the initial and final instant. A graphic representation of the sequence of the different phases for the ascent flight of a rocket is reported in Figure 4.4.

Phase 1: Vertical lift-off. It is performed by the launcher at its initial configuration and it ends at the minimum time needed for a safe clearing of the launch pad. None of the guidance angles is optimizable in this phase. The pitch angle is constantly 90° as the vehicle is lifting off vertically and the yaw angle doesn't have significance.

$$\theta(t) = \frac{\pi}{2}, \quad \psi(t) = \text{insignificant}$$

Phase 2: Pitch push over. The second phase is a pitch over maneuver getting the launcher trajectory out of the vertical direction. The pitch over standard profile is defined as a linear increase of duration

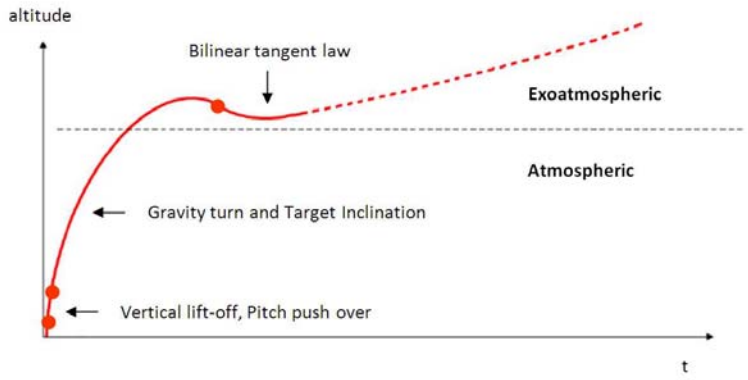


FIGURE 4.4. Launcher flight sequence.

Δt_{PO} and a successive exponential decay

$$\theta(t) = \begin{cases} \gamma(t) - \dfrac{t - t_{F_1}}{\Delta t_{\mathrm{PO}}} \Delta\theta & \text{for } t_{F_1} \leq t \leq t_{F_1} + \Delta t_{\mathrm{PO}} \\ \gamma(t) - \Delta\theta \exp\left(\dfrac{-(t - t_{\mathrm{PO}})}{\Delta t_{\mathrm{PO,decay}}} \right) & \text{for } t_{F_1} + \Delta t_{\mathrm{PO}} < t \leq t_{F_2} \end{cases}$$

$$\psi(t) = \psi_0$$

Δt_{PO} and $\Delta t_{\mathrm{PO,decay}}$ are parameters typical for the launcher vehicle dynamics. Moreover it stands that $t_{\mathrm{PO}} = t_{F_1} + \Delta t_{\mathrm{PO}}$ and $t_{F_2} = t_{F_1} + \Delta t_{\mathrm{PO}} + 3\Delta t_{\mathrm{PO,decay}}$. $\Delta\theta$ and ψ_0 are constant during the phase, and they represent respectively the deviation from the flight-path angle and the initial yaw angle (equal to the heading angle) to allow a final orbit's inclination close to the target one.

Phase 3: Gravity turn and Target inclination. The third phase is performed until the burn out instant of the last lower stage. The pitch angle follows the gravity turn reference law, that corresponds to a zero angle of attack. The pitch angle is defined with a time dependent deviation from its reference law

$$\theta(t) = \gamma(t) + \Delta\theta(t).$$

The reference law of the yaw angle is defined as

$$\tilde{\varphi}(t) = \begin{cases} \arcsin\left(\dfrac{\cos(i_T)}{cos(\delta(t))} \right), & \text{if } -1 \leq \dfrac{\cos(i_T)}{\cos(\delta(t))} \leq 1 \\ \pi/2, & \text{if } \dfrac{\cos(i_T)}{\cos(\delta(t))} > 1 \\ -\pi/2, & \text{if } \dfrac{\cos(i_T)}{\cos(\delta(t))} < -1 \end{cases}$$

where i_T is the target inclination of the final orbit and, as for the pitch angle, the yaw is defined with a deviation from its standard law

$$\varphi(t) = \tilde{\varphi}(t) + \Delta\varphi(t).$$

Phase 4: Bilinear tangent law and Target inclination. The fourth phase is performed by the launcher upper stage. The pitch angle has just three real variables that define its profile

$$\theta(t) = \arctan\left(\frac{a^{\xi}\tan(\theta_0) + (\tan(\theta_f) - a^{\xi}\tan(\theta_0))\hat{t}}{a^{\xi} + (1-a^{\xi})\hat{t}}\right), \quad \hat{t} = \frac{t - t_{F_1}}{t_{F_2} - t_{F_1}}$$

$a \in \mathbb{R} > 1$ constant and θ_0, θ_f, $\xi \in \mathbb{R}$. The yaw angle follows the same law as for the third phase.

Coast phase. The coast phases are performed between the jettison of a stage and the ignition of the following. The engine is switched off and the trajectory propagated subject only to the gravitational force (possibly aerodynamic forces if it is still during the atmospheric flight). The pitch and yaw angles lose significance in the ballistic phase and the only varying parameter is its duration.

4.3. Multiple Shooting Techniques

The attitude angles, as defined in the guidance model, and the thrust level profile are all continuous functions. Only the initial constant values of pitch deviation and yaw during the pitch over maneuver and the three parameters defining the bilinear tangent law are optimizable real values. It is necessary instead to have a parametrization for the pitch and yaw angles during the third and fourth phase (only yaw), and the thrust level profile for all the duration of the flight. A finite number of nodal values is defined on the different phases. The nodal values are optimizable and the full control law is defined through their interpolation. In particular, one node means optimizable constant deviation during the flight, two nodes mean optimizable start and final deviations and interpolation between them and so on. This approach is called direct shooting technique, and belongs to the direct methods for solving optimal control problems. Through the discretization, the infinite dimensional optimization problem is transcribed into an NLP problem, solvable employing efficient algorithms for large and sparse NLP.

The control profiles are not the only discretized functions. To improve the robustness of the trajectory optimization process also the state variables are discretized on the time of flight frame.

The trajectory is not integrated from the beginning to the end, but on shorter intervals. The initial conditions of the integration over each subinterval are optimization variables. Matching constraints for equalizing the values of the optimization variables with the final states coming from the integration on the previous interval are added to the optimization problem formulation. This way, the trajectory is indeed less sensitive to the launch initial conditions.

The ascent trajectory is modeled in different phases, one for each vehicle component. It is a different division of the time of flight with respect to the one given for the definition of the guidance laws. For each phase, a custom number of multiple shooting nodes are chosen and placed homogeneously in

the interval (equally distance one from each other)

$$t_0 < t_1 < ... < t_{n_{MS}} < t_f$$

where t_0 is the launch instant, t_f is the final time of flight (burn out of the upper stage) and n_{MS} is the number of multiple shooting nodes including the time instants connecting the phases. If the number of nodes is zero, the nodes connecting the phases are considered as unique multiple shooting nodes. To each node corresponds a set of additional optimization variables $\{\tilde{x}_i\}_{i=1}^{n_{\mathrm{MS}}}$, $x_i \in \mathbb{R}^n$, one variable for each state in each node, and additional equality constraints $g_i(x(t), u(t), t) = 0$ defined as

$$g_i(x(t), u(t)) = \int_{t_{i-1}}^{t_i} f(x(t), u(t))dt + \tilde{x}_{i-1} - \tilde{x}_i, \quad \text{for } i = 1, ..., n_{MS}$$

where $g : \mathbb{R}^{n_s} \times \mathbb{R}^{n_c} \to \mathbb{R}^{n_s}$, and f represents the right hand side of the EoM. The trajectory is integrated numerically from one multiple shooting node to the next, taking as initial values the current values of the optimization variables.
Additional $n_{MS} \cdot n_s$ optimization variables and equality constraints are added to the original problem. This results in a larger and sparser problem, that can be efficiently solved with the local optimizer WORHP introduced in Section 3.4.3, explicitly declaring the structure of the Jacobian and Hessian matrices.

4.4. Semianalytic Methods

Semianalytic methods for the ascent trajectory integration, intended as definition of a fully analytic solution on subintervals of each component phase, have been derived on the basis of the methodology explained in [**142, 143**]. The procedure has been adapted to the dynamic model in use for achieving, with the optimization, fast payload assessments to be included in the nested trajectory optimization loop, one of the MDO architecture, mentioned in Section 2.4.
The methodology differs between atmospheric and exoatmospheric flights, but the procedure is common. At the time interval $[t_i, t_{i+1}] \subset [t_0, t_f]$ with $t_{i+1} = t_i + \Delta t_i$, the steps are:

Initialization of a set of coefficients at the beginning of each step t_i,
Computation of the propagation step Δt_i,
Computation of the states at the final time step: $x(t_{i+1})$.

During the **atmospheric** flight, the EoM differential equations system is solved for a reduced set of variables: V and γ, assuming the other components as constant, since they are varying at slower rates. The atmospheric part is also a null angle of attack phase, meaning that thrust and velocity are collinear and body frame and relative velocity frame match. The corresponding reduced set of differential equations presented in the dynamic model can

be rewritten highlighting the time dependent variables as

$$\dot{V}(t) = \Gamma + C_1 g \sin\gamma(t) + C_2 g \cos\gamma(t) \tag{4.4.1}$$

$$\dot{\gamma}(t) = C_0 \frac{g\cos\gamma(t)}{V(t)} + C_5 g\cos\gamma(t) + \frac{V(t)\cos\gamma(t)}{r}, \tag{4.4.2}$$

where $\Gamma = {}^{T}/_{m} - KV(t)^2$ and K is the contribution of the aerodynamic force along the direction x_T,

$$K = -\frac{\rho(r) A_{\text{ref}} (C_D(M,\alpha) + C_L(M,\alpha))}{2m},$$

where $g = \mu/r^2$ with $\mu = 398602.0 \cdot 10^9$ the Earth gravitational constant. The terms C_0, C_1, C_2 and C_5 aggregate the terms of the EoM which contain the state variables λ, δ, φ and r, namely (R_E is the Earth radius)

$$\begin{aligned}
C_0 &= C_1 - C_2 \tan\gamma(t), \\
C_1 &= -1 - \frac{3}{2} J_2 \left(\frac{R_E}{r}\right)^2 (1 - 3\sin^2\delta) + \frac{\omega_E^2 r \cos^2\delta}{g}, \\
C_2 &= -3 J_2 \left(\frac{R_E}{r}\right)^2 (\cos\psi \sin\delta \cos\delta) - \frac{\omega_E^2 r \cos\psi \sin\delta \cos\delta}{g}, \\
C_5 &= \frac{2\omega_E \sin\psi \cos\delta}{g\cos\gamma(t)}.
\end{aligned}$$

On an interval $[t_i, t_{i+1}] \subset [t_0, t_f]$ the two states can be expressed as

$$\begin{aligned}
V(t) &= V_i + \Delta V_i(t) \\
\gamma(t) &= \gamma_i + \Delta\gamma_i(t)
\end{aligned}$$

with $t \in [t_i, t_{i+1}]$, $V_i \in \mathbb{R}$ and $\gamma_i \in \mathbb{R}$ the values of the states at the initial instant and $\Delta V_i(t)$, $\Delta\gamma_i(t)$ the state increments on the i-th interval. The objective is to find an analytic expression for the states increments at $t = t_{i+1}$, that means assessing the increment values

$$\Delta V_i(t_{i+1}) := \Delta V_i \text{ and } \Delta\gamma_i(t_{i+1}) := \Delta\gamma_i.$$

The idea of the method is expressing the flight path angle as third order Taylor expansion of the velocity increment

$$\gamma(t_{i+1}) := \gamma_{i+1} \simeq \gamma_i + A\Delta V_i + B\Delta V_i^2 + C\Delta V_i^3 + o(\Delta V_i^4).$$

The coefficients A, B and C need to be determined.
Rewriting the flight path and velocity equations of motion, 4.4.1 and 4.4.2, in terms of the common term ${}^{1}/_{(g\cos\gamma(t))}$ and equaling the equations this yields

$$d\gamma \left(\frac{\Gamma}{g\cos\gamma(t)} + C_1 \tan\gamma(t) + C_2\right) = dV \left(\frac{C_0}{V(t)} + C_5 + \frac{V(t)}{rg}\right)$$

Assuming that just V and γ are varying with the time and that the remaining values are constant over the interval $[t_i, t_{i+1}]$, integrating both sides of the

equation on the given interval it gets that

$$\left(\frac{1+\sin(\gamma_i+\Delta\gamma_i)}{1+\sin\gamma_i}\right)^{\beta}\left(\frac{\cos\gamma_i}{\cos(\gamma_i+\Delta\gamma_i)}\right)^{\beta+C_1} e^{C_2\Delta\gamma_i} = \\ = \left(1+\frac{\Delta V_i}{V_i}\right)^{C_0} e^{\frac{\alpha}{2}\left(2\frac{\Delta V_i}{V_i}+\frac{\Delta V_i^2}{V_i^2}\right)} e^{C_5\Delta V_i}$$

with $\alpha = V_i^2/(gr)$ and $\beta = \Gamma/g$. The two terms of the equation are indicated in the next steps as

$$f_1(\Delta\gamma_i) = g_1(\Delta V_i).$$

Expanding the left-hand-side f_1 in $\Delta\gamma_i$ to first order, $f_1(\Delta\gamma_i) \simeq f_1(0) + f_1'(0)\Delta\gamma_i$ it can be approximated as

$$f_1(\Delta\gamma_i) \simeq 1 + \left(\frac{\beta}{\cos\gamma_i} + C_1\tan\gamma_i + C_2\right)\Delta\gamma_i = 1 + \frac{1}{\omega}\Delta\gamma_i$$

with

$$\omega = \frac{\cos\gamma_i}{\beta + C_1\sin\gamma_i + C_2\cos\gamma_i}.$$

Expanding the right hand side to the third term in ΔV_i,

$$g_1(\Delta V_i) \simeq g_1(0) + g_1'(0)\Delta V_i + g_1''(0)\frac{\Delta V_i^2}{2} + g_1'''(0)\frac{\Delta V_i^3}{6},$$

the Taylor coefficients can be calculated as

$$\begin{aligned} g_1(0) =& 1, \\ g_1'(0) =& \frac{C_0}{V_i} + \eta, \\ g_1''(0) =& \frac{C_0(C_0-1)}{V_i^2} + \frac{2C_0}{V_i}\eta + \eta^2 + \frac{\alpha}{V_i^2}, \\ g_1'''(0) =& \frac{C_0(C_0-1)(C_0-2)}{V_i^3} + 3\frac{C_0(C_0-1)}{V_i^2}\eta + 3\frac{C_0}{V_i}\left(\eta^2+\frac{\alpha}{V_i^2}\right) + \\ & + \eta\left(\eta^2 + \frac{3\alpha}{V_i^2}\right), \end{aligned}$$

with $\eta = {}^{\alpha}/_{V_i} + C_5$. Substituting $\Delta\gamma_i$ with its definition as Taylor expansion in ΔV_i and equaling the two expansions, it is possible to determine the coefficients A, B and C:

$$\begin{aligned} A &:= \omega g'(0), \\ B &:= \omega\frac{g''(0)}{2}, \\ C &:= \omega\frac{g'''(0)}{6}. \end{aligned}$$

Once the increment of the flight path angle has been expressed in terms of the increment of the relative velocity, it is enough to find an analytic expression for the last one. To do this the derivative of the increment is expressed as a

third order Taylor expansion of the increment itself

$$\Delta\dot{V}_i(t) \simeq a + b\Delta V_i(t) + c\Delta V_i^2(t) + d\Delta V_i^3(t) + o(\Delta V_i^4(t)).$$

To determine its coefficients, the equation of motion describing the variation of the relative velocity 4.4.1 is expanded to the second term in $\Delta\gamma_i(t)$

$$\dot{V}(t)_{|[t_i,t_{i+1}]} \simeq \Gamma + C_1 g \left(\sin\gamma_i \left(1 - \frac{(\Delta\gamma_i(t))^2}{2} \right) + \cos\gamma_i \Delta\gamma_i(t) \right) + \\ + C_2 g \left(\cos\gamma_i \left(1 - \frac{(\Delta\gamma_i(t))^2}{2} \right) - \sin\gamma_i \Delta\gamma_i(t) \right).$$

The squared increment of the flight path angle can be approximated as

$$\Delta\gamma_i^2(t) \simeq A^2 \Delta V_i^2(t) + 2AB\Delta V_i^3(t),$$

Combining those informations and standing that

$$\dot{V}(t)_{|[t_i,t_{i+1}]} = [V_i + \Delta V_i(t)]' = \Delta\dot{V}_i(t),$$

it is possible to approximate the derivative of the velocity increment as

$$\begin{aligned} \Delta\dot{V}_i(t) \simeq\, & (\Gamma + C_1 g \sin\gamma_i + C_2 g \cos\gamma_i) + (C_1 g \cos\gamma_i A - \\ & + C_2 g \sin\gamma_i A)\, \Delta V_i(t) + (C_1 g \cos\gamma_i B - C_2 g \sin\gamma_i B - \\ & + \frac{C_1 g A^2 \sin\gamma_i + C_2 g A^2 \cos\gamma_i}{2} \Bigg)\, \Delta V_i^2(t) + (C_1 g \cos\gamma_i C - \\ & + C_1 g AB \sin\gamma_i - C_2 g AB \cos\gamma_i - C_2 g \sin\gamma_i C)\, \Delta V_i^3(t). \end{aligned}$$

Comparing the coefficients of the two differential equations it is possible to determine the analytic expression of a, b, c, d

$$\begin{aligned} a &:= \Gamma + C_1 g \sin\gamma_i + C_2 g \cos\gamma_i, \\ b &:= C_1 g \cos\gamma_i A - C_2 g \sin\gamma_i A, \\ c &:= C_1 g \cos\gamma_i B - C_2 g \sin\gamma_i B - \frac{C_1 g A^2 \sin\gamma_i + C_2 g A^2 \cos\gamma_i}{2}, \\ d &:= C_1 g \cos\gamma_i C - C_1 g AB \sin\gamma_i - C_2 g AB \cos\gamma_i - C_2 g \sin\gamma_i C_\gamma. \end{aligned}$$

It remains now to determine ΔV_i from the differential equation that approximates it. Restricting it to the second order, it results to be a Riccati equation with constant coefficients that can be integrated analytically:

$$\Delta\dot{V}_i(t) = a + b\Delta V_i(t) + c\Delta V_i^2(t),$$

defining the quantities

$$\begin{aligned}\Delta &:= b^2 - 4a,\\ x_1 &:= \frac{-b+\sqrt{\Delta}}{2c},\\ x_2 &:= \frac{-b-\sqrt{\Delta}}{2c},\\ \alpha &:= \sqrt{\frac{a}{c} - \left(\frac{b}{2c}\right)^2},\end{aligned}$$

the solution of the Riccati equation, integrating on the interval $[t_i, t_{i+1}] = [t_i, t_i + \Delta t_i] \subset [t_0, t_f]$, with initial condition $\Delta V_i(t_i) = 0$, is

$$\Delta V_i = \begin{cases} \dfrac{x_1(1 - e^{\sqrt{\Delta}\Delta t_i})}{1 - \frac{x_1 e^{\sqrt{\Delta}\Delta t_i}}{x_2}}, & \Delta \geq 0 \\ -\dfrac{b}{2c} + \alpha\left(\dfrac{\tan(c\alpha\Delta t_i) + \frac{b}{2c\alpha}}{1 + \tan(c\alpha\Delta t_i)\frac{b}{2c\alpha}}\right), & \Delta < 0. \end{cases}$$

Flight path angle and relative velocity are therefore analytically defined on the interval $[t_i, t_{i+1}] \subset [t_0, t_f]$.

It remains now to determine an analytic expression for the remaining state variables r, δ, ϕ and λ.
The derivative of the radius can also be expanded in $\Delta\gamma_i(t)$ starting from the equation of motion

$$\begin{aligned}\dot{r}(t) &= V(t)\sin\gamma(t)\\ \dot{r}(t)_{|[t_i,t_{i+1}]} &\simeq V(t)\left(\sin\gamma_i + \cos\gamma_i\Delta\gamma_i(t) - \frac{\sin\gamma_i}{2}\Delta\gamma_i(t)^2\right)\end{aligned}$$

substituting the approximations over the interval $[t_i, t_{i+1}]$ of $V(t) \simeq V_i + \Delta V_i(t)$ and $\Delta\gamma_i(t) \simeq A\Delta V_i(t) + B\Delta V_i(t)^2$ and integrating on the time interval it yields

$$\begin{aligned}\Delta r_i = \int_{t_i}^{t_{i+1}} \dot{r}(t)\,dt \simeq V_i\sin\gamma_i\Delta t + (AV_i\cos\gamma_i + \sin\gamma_i)\int_{t_i}^{t_{i+1}}\Delta V_i(t)\,dt +\\ + \left(V_iB\cos\gamma_i - \frac{1}{2}V_iA^2\sin\gamma_i + A\cos\gamma_i\right)\int_{t_i}^{t_{i+1}}\Delta V_i^2(t)\,dt.\end{aligned}$$

The analytic expression of ΔV_i is known while its integrals are explicitly computed afterward because they are in common to the approximations of the remaining states.
The remaining equation of motions, related to the three spherical coordinates λ, δ and φ are integrated analytically to compute the states increments making further assumptions on constant terms.

The equation of motion that describes the variation of the declination is

$$\dot{\delta}(t) = \frac{V(t)\cos\gamma(t)\cos\varphi(t)}{r(t)}.$$

Assuming that φ and r are varying at a slow rate, so they can be approximated by their initial values φ_i and r_i, expanding the cos function with Taylor to second order in $\gamma_i(t)$ it gets that

$$\dot{\delta}(t)_{|[t_i,t_{i+1}]} \simeq \frac{V(t)\cos\varphi_i}{r_i}\left(\cos\gamma_i - \sin\gamma_i\Delta\gamma_i(t) - \frac{\cos\gamma_i}{2}\Delta\gamma_i(t)^2\right)$$

Using the same approximations as before and integrating on the time interval it yields

$$\begin{aligned}\Delta\delta_i = \int_{t_i}^{t_{i+1}} \dot{\delta}(t)\,dt \simeq \Bigg(&V_i\cos\gamma_i\Delta t_i + (\cos\gamma_i - V_iA\sin\gamma_i)\int_{t_i}^{t_{i+1}}\Delta V_i(t)\,dt + \\ &-(A\sin\gamma_i + V_iB\sin\gamma_i + \frac{1}{2}V_iA^2\cos\gamma_i)\int_{t_i}^{t_{i+1}}\Delta V_i^2(t)\,dt\Bigg)\frac{\cos\varphi_i}{r_i}.\end{aligned}$$

From the equation of motions the heading angle is varying as

$$\begin{aligned}\dot{\varphi}_i(t) = &\sin\varphi(t)\tan\delta(t)\frac{V(t)\cos\gamma(t)}{r(t)} + \\ &+ 2\omega_E(\sin\delta(t) - \cos\varphi(t)\cos\delta(t)\tan\gamma(t)) + \\ &+ r(t)\omega_E^2\sin\varphi(t)\sin\delta(t)\cos\delta(t)\frac{1}{V(t)\cos\gamma(t)}.\end{aligned}$$

Assuming that only V and γ are varying on the interval $[t_i, t_{i+1}]$, expanding with Taylor in $\Delta\gamma_i(t)$ the trigonometric functions and substituting the approximations of $V(t)$ and $\Delta\gamma_i(t)$ as before, integrating it gets that

$$\begin{aligned}\Delta\varphi_i &= \int_{t_i}^{t_{i+1}}\dot{\varphi}(t)\,dt \\ &\simeq \frac{\sin\varphi_i\tan\delta_i}{r_i}\Bigg(V_i\cos\gamma_i\Delta t_i + (\cos\gamma_i - V_iA\sin\gamma_i)\int_{t_i}^{t_{i+1}}\Delta V_i(t)\,dt - \\ &\quad + \left(A\sin\gamma_i + V_iB\sin\gamma_i + \frac{V_iA^2}{2}\cos\gamma_i\right)\int_{t_i}^{t_{i+1}}\Delta V_i^2(t)\,dt\Bigg) + \\ &\quad + 2\omega_E\left(\sin\delta_i - \cos\varphi_i\cos\delta_i\left(\tan\gamma_i\Delta t_i + \frac{A}{\cos\gamma_i^2}\int_{t_i}^{t_{i+1}}\Delta V_i(t)\,dt\right)\right) + \\ &\quad + r_i\omega_E^2\sin\varphi_i\sin\delta_i\cos\delta_i\left(\frac{1}{V_i\cos\gamma_i}\left(\Delta t_i + \right.\right. \\ &\quad \left.\left. + \frac{A\sin\gamma_iV_i - \cos\gamma_i}{V_i\cos\gamma_i}\int_{t_i}^{t_{i+1}}\Delta V_i(t)\,dt\right)\right).\end{aligned}$$

The longitude variation is described by the equation

$$\dot{\lambda}(t) = \frac{V(t)\cos\gamma(t)\sin\varphi(t)}{\cos\delta(t)r(t)}.$$

Considering φ and δ varying at lower rate and integrating over the interval it gets

$$\Delta\lambda_i = \frac{\sin\varphi_i}{\cos\delta_i}\int_{t_i}^{t_i+\Delta t_i}\frac{V(t)\cos\gamma(t)}{r(t)}\,dt.$$

Knowing that

$$\Delta\delta_i = \cos\varphi_i\int_{t_i}^{t_i+\Delta t_i}\frac{V(t)\cos\gamma(t)}{r(t)}\,dt,$$

it is easy derived that

$$\Delta\lambda_i = \frac{\Delta\delta_i\sin\varphi_i}{\cos\varphi_i\cos\delta_i}.$$

Now it just remains to asses the integrals of the powers of the relative velocity increment

$$\mathcal{I}_n = \int_{t_i}^{t_i+\Delta t_i}\Delta V(t)^n\,dt, \quad n = 1, 2.$$

Its stands the following recursive rule, for $n \geq 1$

$$\begin{aligned}
\mathcal{I}_0 &= \Delta t_i \\
\mathcal{I}_1 &= \frac{1}{2c}\left[-b\Delta t_i + \log\left|\frac{\Delta\dot{V}_i}{a}\right|\right] \\
\mathcal{I}_{n+1} &= \frac{1}{2c}\left[\frac{\Delta V_i^n}{n} - aI_{n-1} - bI_n\right]
\end{aligned}$$

To prove the equation defining $\mathcal{I}_1$ it is enough to prove that

$$-\frac{b}{2c} + \frac{1}{2c}\frac{\Delta\ddot{V}(t)}{\Delta\dot{V}(t)} = \Delta V(t)$$

because

$$\begin{aligned}
\int_{t_i}^{t_i+\Delta t_i} -\frac{b}{2c} + \frac{1}{2c}\frac{\Delta\ddot{V}(t)}{\Delta\dot{V}(t)}\,dt &= -\frac{b}{2c}t|_{t_i}^{t_i+\Delta t_i} + \frac{1}{2c}\ln\Delta\dot{V}(t)|_{t_i}^{t_i+\Delta t_i} \\
&= \frac{1}{2c}\left[-b\Delta t_i + \ln\left|\frac{\Delta\dot{V}_i}{a}\right|\right]
\end{aligned}$$

with $\Delta V(t_i) = 0$ and so $\Delta\dot{V}(t_i) = a$.
This can be done deriving $\Delta\dot{V}(t) = a + b\Delta V(t) + c\Delta V^2(t)$ and obtaining that

$$-\frac{b}{2c} + \frac{b\Delta\dot{V}(t) + 2c\Delta V(t)\Delta\dot{V}(t)}{2c\Delta\dot{V}(t)} = -\frac{b}{2c} + \frac{b}{2c} + \frac{2c\Delta V(t)}{2c} = \Delta V(t).$$

The recursive formula then easily follows.

The timestep used for the integration during the atmospheric flight is computed in such a way that the highest order terms of the Taylor expansions of

the flight path angle and of the derivative of the relative velocity are vanishing. The user has to define two tolerances

$$\begin{aligned} \varepsilon_{acc} &:= |d\Delta V_i^3|, \\ \varepsilon_{\gamma} &:= |C\Delta V_i^3|. \end{aligned}$$

Relative velocity and time increment are in relation as

$$\Delta t_i \approx \frac{\Delta V_i}{|a|},$$

so replacing in it the values of the velocity increment depending on the tolerances is possible to determine two values of the time step

$$\Delta t_1 = \frac{1}{|a|}\left(\frac{\varepsilon_{\mathrm{acc}}}{|d|}\right)^{1/3},\ \Delta t_2 = \frac{1}{|a|}\left(\frac{\varepsilon_{\gamma}}{|C|}\right)^{1/3}.$$

The minimum between the two is used as timestep for the semianalytic derivation

$$\Delta t_i = \{\Delta t_1, \Delta t_2\}.$$

The **exoatmospheric** part is dominated by thrust and gravity forces. The state vector is derived in the ECI reference frame: 3 components are defining the position r and 3 components the absolute velocity. Assuming that on an interval $[t_i, t_i + \Delta t_i]$ the state increment can be written as

$$\begin{aligned} r(t_i + \Delta t_i) &= r_i + \Delta r_i \\ v(t_i + \Delta t_i) &= v_i + \Delta v_i \end{aligned}$$

where $r_i \in \mathbb{R}^3$ and $v_i \in \mathbb{R}^3$ are the initial position and velocity. The objective is to find two functions, f and g such that

$$\begin{aligned} r(t_i + \Delta t_i) &= f(\Delta t_i) r_i + g(\Delta t_i) v_i \\ v(t_i + \Delta t_i) &= \dot{f}(\Delta t_i) r_i + \dot{g}(\Delta t_i) v_i. \end{aligned}$$

Expanding $r(t_i + \Delta t_i)$ up to the third order in Δt_i

$$\begin{aligned} r(t_i + \Delta t_i) &= r(t_i) + \dot{r}(t_i)\Delta t_i + \ddot{r}(t_i)\frac{\Delta t_i^2}{2} + \dddot{r}(t_i)\frac{\Delta t_i^3}{6} \\ &= r_i + v_i \Delta t_i + \ddot{r}(t_i)\frac{\Delta t_i^2}{2} + \dddot{r}(t_i)\frac{\Delta t_i^3}{6}. \end{aligned}$$

With the Newton's second law

$$\ddot{r}(t) = -\frac{\mu}{\|r(t)\|_2^3} \cdot r(t) = -u(t) r(t), \quad t \in [t_0, t_f]$$

differentiating it gives that

$$\begin{aligned} \dddot{r}(t) &= -\dot{u}(t) r(t) - u(t) v(t), \\ r^{(4)}(t) &= (u^2(t) - \ddot{u}(t)) r(t) - 2\dot{u}(t) v(t). \end{aligned}$$

Computing the derivatives of the real function $u(t) = \mu/\|r(t)\|_2^3$, substituting it in the Taylor expansion and grouping similar terms, f and g are defined as

$$\begin{aligned} f(\Delta t_i) &:= 1 - u_i\frac{\Delta t_i^2}{2} - \dot{u}_i\frac{\Delta t_i^3}{6} - (\ddot{u}_i - u_i^2)\frac{\Delta t_i^4}{24}, \\ g(\Delta t_i) &:= \Delta t_i - u_i\frac{\Delta t_i^3}{6} - \dot{u}_i\frac{\Delta t_i^4}{12}, \end{aligned}$$

with derivatives

$$\begin{aligned} \dot{f}(\Delta t_i) &:= -u_i\Delta t_i - \dot{u}_i\frac{\Delta t_i^2}{2} - (\ddot{u}_i - u_i^2)\frac{\Delta t_i^3}{6}, \\ \dot{g}(\Delta t_i) &:= 1 - u_i\frac{\Delta t_i^2}{2} - \dot{u}_i\frac{\Delta t_i^3}{6}. \end{aligned}$$

where $u_i := u(t_i)$, $\dot{u}_i := \dot{u}(t_i)$ and $\ddot{u}_i := \ddot{u}(t_i)$. The position and velocity vectors are then fully described in relation with the gravity force as

$$\begin{aligned} r(t_i + \Delta t_i) &= r_i + v_i\Delta t_i - r_i u_i\frac{\Delta t_i^2}{2} - (r_i\dot{u}_i + v_i u_i)\frac{\Delta t_i^3}{6} - \\ &\quad + ((\ddot{u}_i - u_i^2)r_i + \dot{u}_i v_i)\frac{\Delta t_i^4}{24}, \\ v(t_i + \Delta t_i) &= v_i - u_i r_i\Delta t_i - (\dot{u}_i r_i + u_i v_i)\frac{\Delta t_i^2}{2} - \\ &\quad + ((\ddot{u}_i - u_i^2)r_i + \dot{u}_i v_i)\frac{\Delta t_i^3}{6}. \end{aligned}$$

The next step is to add the thrust contribution. During the exoatmospheric part thrust and velocity vector are not aligned. The two angles that define the launcher attitude with respect to the local frame are the pitch and yaw angles. It is necessary now to rotate the thrust vector defined in the body system as

$$T = \|T\|_2 \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}_{\text{BS}}$$

to the ECI reference frame. The rotation is the product of four rotational matrices. First from the body system it is rotated to the Local Coordinate system using the pitch and yaw angles, then it is transformed in the ECI system using the two angles $\alpha = \pi/2 + \delta$ and $\beta = \text{GST} + \omega t + \lambda$ where GST is the Greenwich sideral time

$$\begin{aligned} T_{\text{V}} &= \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \begin{pmatrix} \|T\|_2 \\ 0 \\ 0 \end{pmatrix}_{\text{BS}} \\ T_{\text{ECI}} &= \begin{bmatrix} \cos(\alpha) & 0 & -\sin(\alpha) \\ 0 & 1 & 0 \\ \sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix} \begin{bmatrix} \cos(\beta) & -\sin(\beta) & 0 \\ \sin(\beta) & \cos(\beta) & 0 \\ 0 & 0 & 1 \end{bmatrix} T_V. \end{aligned}$$

Each component of the rotated thrust can be expanded with Taylor assuming that the evolution of the control angles over an interval is linear that is, being

$\Delta t = t - t_i$ with $t \in [t_i, t_{i+1}]$

$$\begin{aligned} \theta(t) &= \theta(t_i) + \dot{\theta}(t_i)\Delta t \\ \psi(t) &= \psi(t_i) + \dot{\psi}(t_i)\Delta t. \end{aligned}$$

The dynamic in the inertial system is described as

$$\ddot{r}(t) = a_0 + a_1\Delta t + a_2\frac{\Delta t^2}{2} + a_3\frac{\Delta t^3}{6} + o(\Delta t^4),$$

where a_i is the i-th term of the expansion. Integrating it is possible to obtain the contribution of the thrust to the position and velocity vectors increments

$$\begin{aligned} \Delta v_i &= a_0\Delta t_i + a_1\tfrac{\Delta t_i^2}{2} + a_2\tfrac{\Delta t_i^3}{6} + o(\Delta t_i^4), \\ \Delta r_i &= a_0\tfrac{\Delta t_i^2}{2} + a_1\tfrac{\Delta t_i^3}{6} + o(\Delta t_i^4). \end{aligned}$$

Combining thrust and gravity effect the vehicle dynamic can be approximate in the inertial system as

$$\begin{aligned} r(t_i + \Delta t_i) &= r_i + v_i\Delta t_i + (a_0 - u_i r_i)\frac{\Delta t_i^2}{2} + (a_1 - \dot{u}_i r_i - u_i v_i)\frac{\Delta t_i^3}{6}, \\ v(t_i + \Delta t_i) &= v_i + (a_0 - u_i r_i)\Delta t_i + (a_1 - \dot{u}_i r_i - u_i v_i)\frac{\Delta t_i^2}{2} + \\ &\quad + (a_2 - (\ddot{u}_i - u_i^2)r_i - 2\dot{u}_i v_i)\frac{\Delta t_i^3}{6}. \end{aligned}$$

As for the atmospheric sequence the step size is tuned according to the terms of the expansion that are neglected. In particular, assessing the two tolerances ε_r, ε_v on the accuracy of position and velocity norms it gets that

$$\begin{aligned} \Delta t_1 &= \left(\frac{6\varepsilon_r}{\|a_1 - \dot{u}_i r_i - u_i v_i\|_2}\right)^{1/3}, \\ \Delta t_2 &= \left(\frac{6\varepsilon_v}{\|a_2 - (\ddot{u}_i - u_i^2)r_i - 2\dot{u}_i v_i\|_2}\right)^{1/3}. \end{aligned}$$

The step size is chosen as minimum between the two computed values to ensure that precision on both position and velocity are respected

$$\Delta t_i = \min\{\Delta t_1, \Delta t_2\}.$$

This completes the derivation of the analytic equations for the exoatmospheric flight sequence.

CHAPTER 5

Numerical Results

Contents

5.1. Validation of Single Objective MINLP Techniques 110
5.1.1. Analytic Test Problems 110
5.1.2. MDO Test Problem 121
5.2. Validation of Multiple Objective MINLP Techniques 125
5.2.1. Analytic Test Problems 125
5.2.2. MDO Test Problems 153
5.3. Validation of the Trajectory Optimization Subproblem 166
5.3.1. Modeling 167
5.3.2. Test Case 171
5.3.3. Numerical Integration 171
5.3.4. Semianalytic Integration 177
5.4. Comparison of MDO Architectures 180
5.4.1. MDF vs IDF 180
5.4.2. BBO vs NOL 182
5.4.3. Local Refinements 184
5.4.4. Conclusions 185

The MINLP strategies selected for solving the SLO problem presented in Section 3.4 and the different MDO architectures and MDO problem definitions introduced in Section 2.4 have been validated and tested. In the following chapter, the results are discussed and the different techniques are compared. Moreover, the stand-alone ascent trajectory optimization problem is solved with different approaches presented in Sections 4.3 and 4.4 to attain fast payload assessment.
The MINLP algorithms are divided into two groups whether they can handle one or multiple objectives. Their validation procedure follows two steps: first, the validation against a set of benchmark analytic problems taken from literature, and subsequently the validation on applicative MDO problems using as test bench the European Ariane 5 ECA launcher. The algorithms are compared and some conclusions are outlined about the applicability of the methods based on the problem characteristics. The optimal control problem is solved using the sparse NLP solver WORHP, comparing single and multiple

shooting definitions with numeric or semianalytic integration. In the last section, two problem formulations (IDF and MDF) and two MDO architectures (BBO and NOL) are compared on the basis of their computational efficiency and effectiveness on the applicative MDO problem, with the possibility of further improving the global solution by means of local refinements.
All tests have been performed on an Intel Core2 Duo T8100 processor (2.10 GHz, 3 MB L2 Cache), 2048 MB DDR2 RAM.

5.1. Validation of Single Objective MINLP Techniques

The two global approaches for solving single objective MINLPs presented in Section 3.4.1, namely the stochastic PSO algorithm and the deterministic BBWORHP, are tested and compared on single objective analytic and MDO problems.

5.1.1. Analytic Test Problems. The set of MINLP, single objective, benchmark problems is taken from [**144**]. Most of the test problems included in the collection are picked from the GAMS Model Library MINLPlib [**145**]. This is a selection of MINLP problems created by combining small scale models taken from the literature with large industrial models. GAMS provides a C++ implementation through a converter web interface. The remaining problems analyzed in [**144**] have been coded for this test from scratches, since only their Fortran implementation is freely available in the net.
The set of test problems covers a wide variety of different types, from constrained to unconstrained with nonlinearity in objectives and constraints up to a maximum number of 100 integer variables and 73 constraints. All test problems are relaxable. This means, that objective and constraints functions can be evaluated in fractional values and therefore integer variables can be treated as continuous ones. A summary of the problem data is given in Table 5.1, where

- P is the problem identification number and *name* is the identification name used in literature,
- n_x is the total number of optimization variables; n_r is the cardinality of the continuous variables set while the integer variables subset is divided in n_i integer and n_b binary variables,
- m is the number of all constraints divided in m_e equality and m_i inequality constraints,
- f^* is the best known objective value.

P	*name*	n_x	n_r	n_i	n_b	m	m_e	m_i	f^*
1	MITP1	5	2	3	0	1	0	1	$-0.10010 \cdot 10^5$
2	QIP1	4	0	4	0	4	0	4	$-0.20000 \cdot 10^2$
3	MITP2	5	2	0	3	7	0	7	$0.35000 \cdot 10^1$
4	ASAADI11	4	1	3	0	3	0	3	$-0.40957 \cdot 10^2$
5	ASAADI12	4	0	4	0	3	0	3	$-0.38000 \cdot 10^2$
6	ASAADI21	7	3	4	0	4	0	4	$0.69490 \cdot 10^3$

7	ASAADI22	7	0	7	0	4	0	4	$0.70000 \cdot 10^{3}$
8	ASAADI31	10	4	6	0	8	0	8	$0.37220 \cdot 10^{2}$
9	ASAADI32	10	0	10	0	8	0	8	$0.43000 \cdot 10^{2}$
10	DIRTY	25	12	13	0	10	0	10	$-0.30472 \cdot 10^{9}$
11	BRAAK1	7	4	3	0	2	0	2	$0.10000 \cdot 10^{1}$
12	BRAAK2	7	4	3	0	4	0	4	$-0.27183 \cdot 10^{1}$
13	BRAAK3	7	4	3	0	4	0	4	$-0.19656 \cdot 10^{7}$
14	DEX2	2	0	2	0	2	0	2	$-0.56938 \cdot 10^{2}$
15	TP83	5	3	2	0	6	0	6	$-0.30666 \cdot 10^{5}$
16	WP02	2	1	1	0	2	0	2	$-0.24444 \cdot 10^{1}$
17	NVS01	3	1	2	0	3	1	2	$0.12470 \cdot 10^{2}$
18	NVS02	8	3	5	0	3	3	0	$0.59846 \cdot 10^{1}$
19	NVS03	2	0	2	0	2	0	2	$0.16000 \cdot 10^{2}$
20	NVS04	2	0	2	0	0	0	0	$0.72000 \cdot 10^{0}$
21	NVS05	8	6	2	0	9	4	5	$0.54709 \cdot 10^{1}$
22	NVS06	2	0	2	0	0	0	0	$0.17703 \cdot 10^{1}$
23	NVS07	3	0	3	0	2	0	2	$0.40000 \cdot 10^{1}$
24	NVS08	3	1	2	0	3	0	3	$0.23450 \cdot 10^{2}$
25	NVS09	10	0	10	0	0	0	0	$-0.43134 \cdot 10^{2}$
26	NVS10	2	0	2	0	2	0	2	$-0.31080 \cdot 10^{3}$
27	NVS11	3	0	3	0	2	0	2	$-0.43100 \cdot 10^{3}$
28	NVS12	4	0	4	0	4	0	4	$-0.48120 \cdot 10^{3}$
29	NVS13	5	0	5	0	5	0	5	$-0.58520 \cdot 10^{3}$
30	NVS14	8	3	5	0	3	3	0	$-0.40358 \cdot 10^{5}$
31	NVS15	3	0	3	0	1	0	1	$0.10000 \cdot 10^{1}$
32	NVS16	2	0	2	0	0	0	0	$0.70310 \cdot 10^{0}$
33	NVS17	7	0	7	0	7	0	7	$-0.11004 \cdot 10^{4}$
34	NVS18	6	0	6	0	6	0	6	$-0.77840 \cdot 10^{3}$
35	NVS19	8	0	8	0	8	0	8	$-0.10984 \cdot 10^{4}$
36	NVS20	16	11	5	0	8	0	8	$0.23092 \cdot 10^{3}$
37	NVS21	3	1	2	0	2	0	2	$-0.56848 \cdot 10^{1}$
38	NVS22	8	4	4	0	9	4	5	$0.60582 \cdot 10^{1}$
39	NVS23	9	0	9	0	9	0	9	$-0.11252 \cdot 10^{4}$
40	NVS24	10	0	10	0	10	0	10	$-0.10332 \cdot 10^{4}$
41	GEAR	8	4	4	0	4	4	0	0.000
42	GEAR2	28	4	24	0	4	4	0	0.000
43	GEAR3	8	4	4	0	4	4	0	0.000
44	GEAR4	6	2	4	0	1	1	0	$0.16434 \cdot 10^{1}$
45	WINDFAC	14	11	3	0	13	13	0	0.25450
46	SYNTHES1	6	3	0	3	6	0	6	$0.60097 \cdot 10^{1}$
47	SYNTHES2	11	6	0	5	14	0	14	$0.73035 \cdot 10^{2}$
48	SYNTHES3	17	9	0	8	23	2	21	$0.68010 \cdot 10^{2}$
49	FLOUDAS1	5	2	0	3	5	2	3	$0.76672 \cdot 10^{1}$
50	FLOUDAS2	3	2	0	1	3	0	3	$0.10765 \cdot 10^{1}$

51	FLOUDAS3	11	7	0	4	13	0	13	$0.45796 \cdot 10^{1}$
52	FLOUDAS4	11	3	0	8	7	3	4	$-0.09435 \cdot 10^{1}$
53	FLOUDAS5	8	2	6	0	10	0	10	$0.31000 \cdot 10^{2}$
54	FLOUDAS6	5	2	3	0	5	0	5	$-0.17000 \cdot 10^{2}$
55	OAER	9	6	0	3	7	3	4	$-0.19231 \cdot 10^{1}$
56	SPRING	17	5	1	11	8	5	3	0.84620
57	DAKOTA	4	2	2	0	2	0	2	$0.13634 \cdot 10^{1}$
58	PROB02	6	0	6	0	8	0	8	$0.11224 \cdot 10^{6}$
59	PROB03	2	0	2	0	1	0	1	$0.10000 \cdot 10^{2}$
60	PROB10	2	1	1	0	2	0	2	$0.34455 \cdot 10^{1}$
61	BATCH	47	23	0	24	73	12	61	$0.28551 \cdot 10^{6}$
62	BATCHDES	19	10	0	9	19	6	13	$0.16743 \cdot 10^{6}$
63	DU_OPT5	20	7	13	0	9	0	9	$0.80737 \cdot 10^{1}$
64	DU_OPT	20	7	13	0	9	0	9	$0.35563 \cdot 10^{1}$
65	ST_E13	2	1	0	1	2	0	2	$0.20000 \cdot 10^{1}$
66	ST_E32	35	16	19	0	18	17	1	$-0.14304 \cdot 10^{1}$
67	ST_E36	2	1	1	0	2	1	1	$-0.24600 \cdot 10^{3}$
68	ST_E38	4	2	2	0	3	0	3	$0.71977 \cdot 10^{4}$
69	ST_MIQ1	5	0	0	5	1	0	1	$0.28100 \cdot 10^{3}$
70	ST_MIQ2	4	0	4	0	3	0	3	$0.20000 \cdot 10^{1}$
71	ST_MIQ3	2	0	2	0	1	0	1	$-0.60000 \cdot 10^{1}$
72	ST_MIQ4	6	3	0	3	4	0	4	$-0.45740 \cdot 10^{4}$
73	ST_MIQ5	7	5	2	0	13	0	13	$-0.33389 \cdot 10^{3}$
74	ST_TEST1	5	0	5	0	1	0	1	0.0000
75	ST_TEST2	6	0	6	0	2	0	2	$-0.92500 \cdot 10^{1}$
76	ST_TEST3	13	0	13	0	10	0	10	$-0.70000 \cdot 10^{1}$
77	ST_TEST4	6	0	6	0	5	0	5	$-0.70000 \cdot 10^{1}$
78	ST_TEST5	10	0	10	0	11	0	11	$-0.11000 \cdot 10^{3}$
79	ST_TEST6	10	0	0	10	5	0	5	$0.47100 \cdot 10^{3}$
80	ST_TEST8	24	0	24	0	20	0	20	$-0.29605 \cdot 10^{5}$
81	ST_TESTGR1	10	0	10	0	5	0	5	$-0.12812 \cdot 10^{2}$
82	ST_TESTGR3	20	0	20	0	20	0	20	$-0.20590 \cdot 10^{2}$
83	ST_TESTPH4	3	0	3	0	10	0	10	$-0.80500 \cdot 10^{2}$
84	TLN2	8	0	6	2	12	0	12	$0.53000 \cdot 10^{1}$
85	TLN4	24	0	20	4	24	0	24	$0.83000 \cdot 10^{1}$
86	TLN5	35	0	30	5	30	0	30	$0.10300 \cdot 10^{2}$
87	TLN6	48	0	42	6	36	0	36	$0.15300 \cdot 10^{2}$
88	PROCSEL	10	7	0	3	7	4	3	$-0.19231 \cdot 10^{1}$
89	TLOSS	48	0	42	6	53	0	53	$0.16300 \cdot 10^{2}$
90	TLTR	48	0	36	12	54	0	54	$0.48067 \cdot 10^{2}$
91	ALAN	8	4	0	4	7	2	5	$0.29250 \cdot 10^{1}$
92	MEANVARX	35	21	0	14	44	8	36	$0.14369 \cdot 10^{2}$
93	HMITTELMANN	16	0	0	16	7	0	7	$0.13000 \cdot 10^{2}$
94	MIP_EX	5	2	0	3	7	0	7	$0.35000 \cdot 10^{1}$

95	MGRID_CYCLE1	5	0	5	0	1	0	1	$0.80000 \cdot 10^1$
96	MGRID_CYCLE2	10	0	10	0	1	0	1	$0.30000 \cdot 10^3$
97	CROP5	5	0	5	0	3	0	3	0.10041
98	CROP20	20	0	20	0	3	0	3	0.12456
99	CROP50	50	0	50	0	3	0	3	0.32424
100	CROP100	100	0	100	0	3	0	3	0.85147

TABLE 5.1. Mixed integer test problems definition.

The problems that belong to the MINLPlib collection provide analytically the Jacobian matrix and the gradient of the objective. Hence the sparsity structure of each problem can be exploited inside the BBWORHP algorithm. Indeed, the method makes use of the sparse NLP solver WORHP to solve the relaxed optimization problem at each node of the decision tree (see Section 3.4.1.2 for the algorithm details). In the stochastic strategy, since the algorithm doesn't make use of gradient information, no knowledge about derivatives or structures is required.
A good initial guess is provided in the source code of every test problem. For the BBWORHP algorithm the choice of the initial guess is crucial for the performance of the method. Although the branch and bound technique has a global search capability with respect to the integer variables, the solution of the relaxed NLP problem at each node of the tree makes use of local techniques, hence only local convergence is guaranteed. For the PSO algorithm instead, only one of the particles in the swarm is initialized with the given initial guess while the rest is initialized randomly. Therefore the robustness of the evolutionary strategy, intended as the standard deviation of the objective values obtained from different initial swarms, needs to be evaluated.
In the BBWORHP algorithm the problem functions are scaled externally at each node of the tree with the values computed in the initial solution. Indicating by $x_{0,i}$ the initial guess of the sub-problem solved in the i-th node of the decision tree, the value of objective and constraints are scaled respectively by the factors:

$$\bar{f} = |f(x_{0,i})|, \qquad \bar{c} = \|c(x_{0,i})\|_2,$$

with $f : \mathbb{R}^{n_x} \to \mathbb{R}$ and $c : \mathbb{R}^{n_x} \to \mathbb{R}^m$ the objective and constraint functions. Constraints bounds and tolerances are also scaled according to these factors. The default setting for the parameters of the two optimization strategies are reported in Table 5.2. For the branch and bound algorithm, the first subproblem from its active set is selected and branched respect to the variable with the highest fractional part (these refers to branch and selection rules equal to 1). For the remaining parameters the definitions of Section 3.4.1 apply. The results obtained with the BBWORHP strategy are listed in Table 5.3. Please note that 0.0 represents a value smaller than the machine precision. They include the number of function evaluations (feval), the number of NLP sub-problems solved (npb), the optimal value found ($\widetilde{f}^*$), the error of the objective value attained with respect to the best known value (err) and the

BBWORHP parameters		PSO parameters	
Branch rule	1	Swarm size	500
Selection rule	1	Maxiter	1000
Optimality tolerance	10^{-6}	Initial inertia	0.5
Feasibility tolerance	10^{-6}	Final inertia	0.5
Gradient of f user defined	true	Self confidence	1.0
Jacobian of g user defined	true	Swarm confidence	2.0
Maxiter	1000	Mutation probability	0.01
		Max velocity	1.0

TABLE 5.2. Default parameters for single objective strategies.

constraints violation of the final solution (vio). The error on the objective value is computed as the difference between the optimal value found by the method and the best known one, f^* , scaled with the absolute value of the last one

$$\text{err} = \frac{\widetilde{f}^* - f^*}{|f^*|}.$$

In this way a negative value of the error means that the algorithm was able to find an improved solution. This improvement is strictly related to the optimality and feasibility tolerances used in the test cases.

	BBWORHP				
P	feval	npb	$\widetilde{f}^*$	err	vio
1	3072	11	$-0.10010\cdot10^{5}$	$0.996\cdot10^{-8}$	0.000
2	456	7	$-0.20000\cdot10^{2}$	$0.658\cdot10^{-11}$	0.000
3	1114	7	$0.35000\cdot10^{1}$	$-0.306\cdot10^{-11}$	$0.164\cdot10^{-10}$
4	460	3	$-0.40957\cdot10^{2}$	$-0.200\cdot10^{-4}$	$0.144\cdot10^{-7}$
5	1686	7	$-0.38000\cdot10^{2}$	$0.127\cdot10^{-10}$	0.000
6	3662	9	$0.69490\cdot10^{3}$	$0.344\cdot10^{-6}$	0.000
7	8946	31	$0.70000\cdot10^{3}$	$0.222\cdot10^{-13}$	0.000
8	11957	21	$0.37219\cdot10^{2}$	$-0.139\cdot10^{-4}$	$0.120\cdot10^{-9}$
9	51817	129	$0.43000\cdot10^{2}$	0.000	0.000
10	40098	61	$-0.30353\cdot10^{9}$	$0.391\cdot10^{-2}$	0.000
11	8906	7	$0.10000\cdot10^{1}$	$0.720\cdot10^{-8}$	0.000
12	19140	9	$-0.27183\cdot10^{1}$	$-0.228\cdot10^{-6}$	0.000
13	3335	5	$-0.19656\cdot10^{7}$	$0.652\cdot10^{-5}$	0.000
14	952	7	$-0.56938\cdot10^{2}$	0.000	0.000
15	994	3	$-0.30666\cdot10^{5}$	$0.152\cdot10^{-8}$	$0.114\cdot10^{-9}$
16	983	3	$-0.24444\cdot10^{1}$	$-0.182\cdot10^{-5}$	0.000
17	9522	7	$0.12470\cdot10^{2}$	$-0.250\cdot10^{-5}$	$0.424\cdot10^{-8}$
18	728	37	$0.59642\cdot10^{1}$	$-0.341\cdot10^{-2}$	$0.298\cdot10^{-11}$
19	639	9	$0.16000\cdot10^{2}$	0.000	0.000

20	674	15	$0.72000 \cdot 10^{0}$	$0.262 \cdot 10^{-9}$	0.000
21	2524	5	$0.54362 \cdot 10^{1}$	$-0.634 \cdot 10^{-2}$	$0.952 \cdot 10^{-8}$
22	43	5	$0.17703 \cdot 10^{1}$	$0.706 \cdot 10^{-5}$	0.000
23	538	5	$0.40000 \cdot 10^{1}$	0.000	0.000
24	89	7	$0.23450 \cdot 10^{2}$	$0.117 \cdot 10^{-5}$	$0.278 \cdot 10^{-7}$
25	145	3	$-0.43134 \cdot 10^{2}$	$-0.856 \cdot 10^{-6}$	0.000
26	22	5	$-0.31080 \cdot 10^{3}$	$-0.183 \cdot 10^{-15}$	0.000
27	275	7	$-0.43100 \cdot 10^{3}$	$0.132 \cdot 10^{-15}$	0.000
28	465	9	$-0.48120 \cdot 10^{3}$	0.000	0.000
29	693	13	$-0.58520 \cdot 10^{3}$	0.000	0.000
30	1489	21	$-0.40358 \cdot 10^{5}$	$0.761 \cdot 10^{-9}$	$0.175 \cdot 10^{-7}$
31	1260	11	$0.10000 \cdot 10^{1}$	0.000	0.000
32	26	5	0.70313	$0.356 \cdot 10^{-4}$	0.000
33	756	33	$-0.11004 \cdot 10^{4}$	$0.248 \cdot 10^{-14}$	0.000
34	370	31	$-0.77840 \cdot 10^{3}$	0.000	0.000
35	966	55	$-0.10984 \cdot 10^{4}$	0.000	0.000
36	161	17	$0.23092 \cdot 10^{3}$	$0.683 \cdot 10^{-6}$	0.000
37	874	17	$-0.56848 \cdot 10^{1}$	$0.308 \cdot 10^{-5}$	$0.479 \cdot 10^{-9}$
38	3980	9	$0.60582 \cdot 10^{1}$	$0.330 \cdot 10^{-5}$	$0.274 \cdot 10^{-5}$
39	2195	81	$-0.11252 \cdot 10^{4}$	$0.202 \cdot 10^{-15}$	0.000
40	2198	87	$-0.10332 \cdot 10^{4}$	$0.831 \cdot 10^{-12}$	0.000
41	967	13	$0.77786 \cdot 10^{-6}$	$0.77786 \cdot 10^{-6}$	0.000
42	121949	141	$0.13354 \cdot 10^{-4}$	$0.13354 \cdot 10^{-4}$	$0.476 \cdot 10^{-9}$
43	72	13	$0.77786 \cdot 10^{-6}$	$0.77786 \cdot 10^{-6}$	$0.194 \cdot 10^{-9}$
44	No integer solution found				
45	901	11	0.25449	$-0.499 \cdot 10^{-4}$	$0.110 \cdot 10^{-11}$
46	40	5	$0.600983 \cdot 10^{1}$	$0.286 \cdot 10^{-5}$	$0.385 \cdot 10^{-8}$
47	71	9	$0.73035 \cdot 10^{2}$	$0.172 \cdot 10^{-6}$	$0.118 \cdot 10^{-12}$
48	160	15	$0.68010 \cdot 10^{2}$	$0.644 \cdot 10^{-6}$	$0.755 \cdot 10^{-11}$
49	10	1	$0.76672 \cdot 10^{1}$	$-0.394 \cdot 10^{-8}$	$0.553 \cdot 10^{-9}$
50	1821	3	$0.10765 \cdot 10^{1}$	$0.399 \cdot 10^{-4}$	$0.420 \cdot 10^{-7}$
51	59	7	$0.45796 \cdot 10^{1}$	$-0.382 \cdot 10^{-5}$	0.000
52	907	11	-0.94347	$0.313 \cdot 10^{-4}$	$0.517 \cdot 10^{-7}$
53	1061	5	$0.31000 \cdot 10^{2}$	$-0.268 \cdot 10^{-10}$	$0.420 \cdot 10^{-9}$
54	166	1	$-0.17000 \cdot 10^{2}$	$-0.434 \cdot 10^{-7}$	$0.125 \cdot 10^{-6}$
55	292	5	$-0.19231 \cdot 10^{1}$	$0.939 \cdot 10^{-6}$	$0.860 \cdot 10^{-9}$
56	27855	13	0.84625	$0.540 \cdot 10^{-4}$	$0.155 \cdot 10^{-10}$
57	239	5	$0.13634 \cdot 10^{1}$	$0.145 \cdot 10^{-10}$	0.000
58	807	7	$0.11223 \cdot 10^{6}$	$-0.170 \cdot 10^{-12}$	$0.512 \cdot 10^{-10}$
59	429	9	$0.10000 \cdot 10^{2}$	0.000	0.000
60	574	5	$0.34455 \cdot 10^{1}$	$0.114 \cdot 10^{-5}$	0.000
61	6545	17	$0.28551 \cdot 10^{6}$	$0.216 \cdot 10^{-9}$	$0.120 \cdot 10^{-9}$
62	11470	5	$0.16743 \cdot 10^{6}$	$0.249 \cdot 10^{-10}$	$0.287 \cdot 10^{-9}$
63	36650	67	$0.85790 \cdot 10^{1}$	$0.626 \cdot 10^{-1}$	0.000

64	31562	43	$0.35568 \cdot 10^{1}$	$0.138 \cdot 10^{-3}$	$0.573 \cdot 10^{-8}$
65	151	3	$0.20000 \cdot 10^{1}$	$0.699 \cdot 10^{-11}$	0.000
66	21218	25	$-0.14304 \cdot 10^{1}$	$-0.509 \cdot 10^{-5}$	$0.217 \cdot 10^{-7}$
67	78	5	$-0.24600 \cdot 10^{3}$	$0.820 \cdot 10^{-14}$	$0.160 \cdot 10^{-7}$
68	729	3	$0.71977 \cdot 10^{4}$	$0.666 \cdot 10^{-8}$	$0.152 \cdot 10^{-8}$
69	2504	11	$0.28100 \cdot 10^{3}$	$0.437 \cdot 10^{-11}$	0.000
70	64	9	$0.20000 \cdot 10^{1}$	$-0.711 \cdot 10^{-14}$	$0.178 \cdot 10^{-14}$
71	5	1	$-0.60000 \cdot 10^{1}$	$0.194 \cdot 10^{-12}$	0.000
72	21	3	$-0.45740 \cdot 10^{4}$	$0.859 \cdot 10^{-13}$	$0.264 \cdot 10^{-11}$
73	146	1	$-0.33389 \cdot 10^{3}$	$0.333 \cdot 10^{-7}$	$0.506 \cdot 10^{-10}$
74	36	15	$0.38068 \cdot 10^{-8}$	$0.38068 \cdot 10^{-8}$	0.000
75	19	5	$-0.92500 \cdot 10^{1}$	$0.655 \cdot 10^{-8}$	0.000
76	369	11	$-0.70000 \cdot 10^{1}$	$-0.215 \cdot 10^{-8}$	$0.184 \cdot 10^{-7}$
77	24	3	$-0.70000 \cdot 10^{1}$	$-0.106 \cdot 10^{-11}$	$0.149 \cdot 10^{-10}$
78	41099	29	$-0.11000 \cdot 10^{3}$	$0.108 \cdot 10^{-8}$	0.000
79	8530	33	$0.47100 \cdot 10^{3}$	$-0.12679 \cdot 10^{-8}$	$0.349 \cdot 10^{-7}$
80	450	7	$-0.29605 \cdot 10^{5}$	$0.844 \cdot 10^{-12}$	$0.220 \cdot 10^{-10}$
81	2330	49	$-0.12812 \cdot 10^{2}$	$0.232 \cdot 10^{-10}$	$0.119 \cdot 10^{-10}$
82	No integer solution found				
83	266	7	$-0.80500 \cdot 10^{2}$	0.000	0.000
84	26249	83	$0.53000 \cdot 10^{1}$	$-0.279 \cdot 10^{-10}$	$0.776 \cdot 10^{-9}$
85	No integer solution found				
86	No integer solution found				
87	No integer solution found				
88	56	9	$-0.19231 \cdot 10^{1}$	$0.653 \cdot 10^{-6}$	$0.363 \cdot 10^{-8}$
89	59276	85	$0.16300 \cdot 10^{2}$	$0.125 \cdot 10^{-11}$	$0.819 \cdot 10^{-8}$
90	208981	91	$0.48066 \cdot 10^{2}$	$-0.693 \cdot 10^{-6}$	$0.325 \cdot 10^{-11}$
91	1117	9	$0.29250 \cdot 10^{1}$	$0.948 \cdot 10^{-11}$	$0.756 \cdot 10^{-10}$
92	44	9	$0.14369 \cdot 10^{2}$	$0.224 \cdot 10^{-5}$	$0.128 \cdot 10^{-10}$
93	30180	201	$0.13000 \cdot 10^{2}$	$-0.178 \cdot 10^{-14}$	$0.239 \cdot 10^{-13}$
94	1294	7	$0.35000 \cdot 10^{1}$	$0.585 \cdot 10^{-12}$	$0.417 \cdot 10^{-12}$
95	2884	23	$0.80000 \cdot 10^{1}$	$0.704 \cdot 10^{-12}$	0.000
96	80172	201	$0.31800 \cdot 10^{3}$	$0.600 \cdot 10^{-1}$	0.000
97	3802	21	0.10041	$0.122 \cdot 10^{-7}$	0.000
98	1138591	1159	0.11166	-0.103	0.000
99	No integer solution found				
100	No integer solution found				

TABLE 5.3. Results of the MINLP test problems solved with the deterministic BBWORHP strategy. 93 problem solved with an average error of 0.0026.

To reduce the stochastic effect in the particle swarm algorithm, 20 runs with different seeds for the random number generator are performed for each test

problem. The best solution over the 20 runs is returned as the optimal solution found.
The PSO algorithm updates the particle position with the dynamic equations described in Section 3.4.1.1. If the particle position vector is assuming integer value along one dimension, after the move the new value of the position vector along the same dimension is not assured to be integer. If the dimension in the search space corresponds to a discrete design variable, the real value returned by the algorithm needs to be rounded before evaluating the fitness function. Naming $x_i \in \mathbb{Z}$ the integer component of the position vector $X = (x_1, x_2, ..., x_{n_x}) \in \mathbb{R}^{n_x}$ and $\bar{x}_i$ the corresponding real value returned by the algorithm, the mapping is defined as

$$x_i = \begin{cases} \lfloor \bar{x}_i \rfloor & \text{if } \bar{x}_i - \lfloor \bar{x}_i \rfloor < 0.5 \\ \lceil \bar{x}_i \rceil & \text{otherwise.} \end{cases}$$

Moreover, the objective function is scaled by the value $|f^*| \in \mathbb{R}$ and the constraints are relaxed by the factor 10^{-6} to match the settings of the previous strategy.
In Table 5.4 the results are listed, reporting for each test problem the number of function evaluations (feval) necessary to converge according to the average improvement stopping criteria discussed in Section 3.4.1.1 with number of compared generations equal to 20 and deviation from the self best fitness smaller than 10^{-6}, the number of feasible solutions obtained in the different runs (n_f), the best optimal value found ($\widetilde{f}^*$), the standard deviation of the feasible set of solutions from the objective mean value (std) and the error from the best known optimal value (err) and constraint violation (vio) of the best solution attained.

	PSO					
P	feval	n_f	$\widetilde{f}^*$	std	err	vio
1	387000	20	$-0.10010 \cdot 10^5$	0.00	0.00	0.00
2	18000	20	$-0.20000 \cdot 10^2$	0.00	0.00	0.00
3	103000	20	$0.35000 \cdot 10^1$	0.46	0.00	0.00
4	54500	20	$-0.40957 \cdot 10^2$	$0.15 \cdot 10^{-13}$	$-0.20 \cdot 10^{-4}$	0.00
5	54000	20	$-0.38000 \cdot 10^2$	0.00	0.00	0.00
6	25500	20	$0.69490 \cdot 10^3$	$0.24 \cdot 10^{12}$	$0.37 \cdot 10^{-8}$	0.00
7	23500	20	$0.70000 \cdot 10^3$	0.00	0.00	0.00
8	337000	20	$0.37221 \cdot 10^2$	$0.24 \cdot 10^2$	$0.46 \cdot 10^{-4}$	0.00
9	46000	20	$0.43000 \cdot 10^2$	$0.76 \cdot 10^1$	0.00	0.00
10	218500	20	$-0.30283 \cdot 10^9$	$0.85 \cdot 10^6$	$0.62 \cdot 10^{-2}$	0.00
11	89500	20	$0.10000 \cdot 10^1$	$0.88 \cdot 10^{-16}$	0.00	0.00
12	63000	20	$-0.27183 \cdot 10^1$	$0.97 \cdot 10^{-15}$	$-0.27 \cdot 10^{-6}$	0.00
13	no feasible solution found					
14	43000	20	$-0.56937 \cdot 10^2$	0.00	0.00	0.00
15	90000	20	$-0.30666 \cdot 10^5$	$0.59 \cdot 10^2$	$-0.27 \cdot 10^{-11}$	0.00

16	21000	20	$-0.24444 \cdot 10^{1}$	$0.91 \cdot 10^{-15}$	$-0.18 \cdot 10^{-5}$	0.00
17	490000	20	$0.15458 \cdot 10^{2}$	$0.84 \cdot 10^{2}$	0.24	0.00
18	384500	11	$0.64717 \cdot 10^{1}$	0.9	$0.81 \cdot 10^{-1}$	$0.12 \cdot 10^{-13}$
19	421000	20	$0.16000 \cdot 10^{2}$	0.00	0.00	0.00
20	390500	20	0.72000	$0.10 \cdot 10^{2}$	$0.85 \cdot 10^{-14}$	0.00
21	498500	16	$0.32171 \cdot 10^{2}$	$0.31 \cdot 10^{3}$	$0.49 \cdot 10^{1}$	0.00
22	378000	20	$0.17703 \cdot 10^{1}$	$0.23 \cdot 10^{-15}$	$0.71 \cdot 10^{-5}$	0.00
23	500500	20	$0.40000 \cdot 10^{1}$	$0.36 \cdot 10^{1}$	0.00	0.00
24	500500	20	$0.23450 \cdot 10^{2}$	$0.11 \cdot 10^{1}$	$0.12 \cdot 10^{-5}$	0.00
25	29500	20	$-0.43134 \cdot 10^{2}$	$0.20 \cdot 10^{1}$	$-0.86 \cdot 10^{-6}$	0.00
26	421000	20	$-0.31080 \cdot 10^{3}$	0.74	$-0.18 \cdot 10^{-15}$	0.00
27	500500	20	$-0.43100 \cdot 10^{3}$	0.00	0.00	0.00
28	500500	20	$-0.48120 \cdot 10^{3}$	0.00	0.00	0.00
29	500500	20	$-0.58520 \cdot 10^{3}$	$0.68 \cdot 10^{2}$	0.00	0.00
30	369500	11	$-0.31939 \cdot 10^{5}$	$0.12 \cdot 10^{5}$	0.21	0.00
31	500500	20	$0.10000 \cdot 10^{1}$	0.00	0.00	0.00
32	394500	20	0.70312	$0.51 \cdot 10^{1}$	$0.36 \cdot 10^{-4}$	0.00
33	500500	15	$-0.11004 \cdot 10^{4}$	$0.15 \cdot 10^{3}$	$-0.21 \cdot 10^{-15}$	0.00
34	500500	14	$-0.77840 \cdot 10^{3}$	$0.52 \cdot 10^{1}$	0.00	0.00
35	500500	10	$-0.10984 \cdot 10^{4}$	$0.25 \cdot 10^{3}$	0.00	0.00
36	500500	20	$0.22330 \cdot 10^{4}$	$0.24 \cdot 10^{9}$	$0.87 \cdot 10^{1}$	0.00
37	382500	20	$-0.56848 \cdot 10^{1}$	0.57	$0.31 \cdot 10^{-5}$	0.00
38	no feasible solution found					
39	500500	8	$-0.11252 \cdot 10^{4}$	$0.23 \cdot 10^{2}$	$0.20 \cdot 10^{-15}$	0.00
40	500500	3	$-0.10312 \cdot 10^{4}$	$0.65 \cdot 10^{2}$	$0.19 \cdot 10^{-2}$	0.00
41	79000	20	$0.23078 \cdot 10^{-10}$	$0.17 \cdot 10^{-8}$	$0.23 \cdot 10^{-10}$	0.00
42	121000	20	$0.35763 \cdot 10^{-2}$	$0.13 \cdot 10^{2}$	$0.36 \cdot 10^{-2}$	0.00
43	119000	20	$0.34909 \cdot 10^{-4}$	$0.28 \cdot 10^{1}$	$0.35 \cdot 10^{-4}$	0.00
44	416000	6	$0.16434 \cdot 10^{1}$	$0.22 \cdot 10^{-4}$	$0.17 \cdot 10^{-4}$	0.00
45	no feasible solution found					
46	500500	20	$0.60098 \cdot 10^{1}$	0.58	$0.29 \cdot 10^{-5}$	0.00
47	263500	20	$0.73035 \cdot 10^{2}$	$0.73 \cdot 10^{1}$	$0.17 \cdot 10^{-6}$	0.00
48	69000	18	$0.77247 \cdot 10^{2}$	$0.22 \cdot 10^{2}$	0.14	0.00
49	61500	20	$0.76672 \cdot 10^{1}$	0.26	$-0.41 \cdot 10^{-8}$	$0.22 \cdot 10^{-15}$
50	122500	20	$0.10765 \cdot 10^{1}$	0.00	$0.40 \cdot 10^{-4}$	0.00
51	123500	20	$0.52727 \cdot 10^{1}$	0.99	0.15	0.00
52	500500	20	-0.83050	0.21	0.12	$0.33 \cdot 10^{-14}$
53	46500	17	$0.31000 \cdot 10^{2}$	$0.50 \cdot 10^{1}$	0.00	$0.44 \cdot 10^{-15}$
54	193000	20	$-0.17000 \cdot 10^{2}$	$0.59 \cdot 10^{1}$	0.00	0.00
55	500500	14	$-0.19169 \cdot 10^{1}$	$0.15 \cdot 10^{1}$	$0.32 \cdot 10^{-2}$	0.00
56	no feasible solution found					
57	105500	20	$0.13634 \cdot 10^{1}$	$0.64 \cdot 10^{-1}$	$-0.33 \cdot 10^{-15}$	0.00
58	389000	9	$0.23200 \cdot 10^{6}$	$0.12 \cdot 10^{6}$	$0.11 \cdot 10^{1}$	0.00
59	17000	20	$0.10000 \cdot 10^{2}$	0.00	0.00	0.00

60	55000	20	$0.34455 \cdot 10^{1}$	$0.91 \cdot 10^{-15}$	$0.11 \cdot 10^{-5}$	0.00
61	no feasible solution found					
62	192500	13	$0.18120 \cdot 10^{6}$	$0.45 \cdot 10^{5}$	$0.82 \cdot 10^{-1}$	0.00
63	165500	20	$0.13515 \cdot 10^{2}$	$0.21 \cdot 10^{2}$	0.67	0.00
64	500500	20	$0.48839 \cdot 10^{1}$	$0.60 \cdot 10^{1}$	0.37	0.00
65	57500	20	$0.20000 \cdot 10^{1}$	$0.23 \cdot 10^{-15}$	$-0.11 \cdot 10^{-15}$	0.00
66	no feasible solution found					
67	36000	20	$-0.16644 \cdot 10^{3}$	$0.45 \cdot 10^{-10}$	0.32	$0.22 \cdot 10^{-21}$
68	290500	20	$0.71977 \cdot 10^{4}$	$0.55 \cdot 10^{2}$	$0.67 \cdot 10^{-8}$	0.00
69	17000	20	$0.28100 \cdot 10^{3}$	0.00	0.00	0.00
70	no feasible solution found					
71	500500	5	$0.60000 \cdot 10^{1}$	$0.99 \cdot 10^{1}$	$0.20 \cdot 10^{1}$	0.00
72	no feasible solution found					
73	355500	12	$-0.22550 \cdot 10^{3}$	$0.28 \cdot 10^{4}$	0.32	0.00
74	500500	20	0.00	0.00	0.00	0.00
75	no feasible solution found					
76	no feasible solution found					
77	no feasible solution found					
78	54500	20	$-0.11000 \cdot 10^{3}$	0.00	0.00	0.00
79	43500	20	$0.47100 \cdot 10^{3}$	0.00	0.00	0.00
80	no feasible solution found					
81	437500	18	$-0.12771 \cdot 10^{2}$	$0.41 \cdot 10^{1}$	$0.32 \cdot 10^{-2}$	0.00
82	no feasible solution found					
83	317500	20	$-0.80500 \cdot 10^{2}$	0.00	0.00	0.00
84	146500	17	$0.53000 \cdot 10^{1}$	$0.92 \cdot 10^{-15}$	0.00	0.00
85	55000	17	$0.12000 \cdot 10^{2}$	$0.30 \cdot 10^{1}$	0.45	0.00
86	109000	7	$0.16900 \cdot 10^{2}$	$0.70 \cdot 10^{1}$	0.64	0.00
87	no feasible solution found					
88	500500	11	$-0.18975 \cdot 10^{1}$	$0.19 \cdot 10^{1}$	$0.13 \cdot 10^{-1}$	0.00
89	no feasible solution found					
90	no feasible solution found					
91	455500	8	$0.29250 \cdot 10^{1}$	0.46	$-0.18 \cdot 10^{-14}$	0.00
92	500500	3	$0.19713 \cdot 10^{2}$	$0.34 \cdot 10^{1}$	0.37	$0.57 \cdot 10^{-13}$
93	17000	20	$0.13000 \cdot 10^{2}$	0.00	0.00	0.00
94	28000	20	$0.35000 \cdot 10^{1}$	0.00	0.00	0.00
95	35500	20	$0.80000 \cdot 10^{1}$	0.73	0.00	0.00
96	91000	20	$0.30400 \cdot 10^{3}$	$0.26 \cdot 10^{2}$	$0.13 \cdot 10^{-1}$	0.00
97	32500	20	0.10041	$0.28 \cdot 10^{-16}$	$0.12 \cdot 10^{-7}$	0.00
98	35500	20	0.11170	$0.33 \cdot 10^{-1}$	-0.10	0.00
99	47000	20	0.48916	0.14	0.51	0.00
100	228500	19	$0.17380 \cdot 10^{1}$	0.74	$0.10 \cdot 10^{1}$	0.00

TABLE 5.4. Results of the MINLP test problems solved with the stochastic PSO strategy. 85 problem solved with an average error of 0.2681.

5.1.1.1. *Conclusions.* Two global single objective optimization strategies for solving mixed integer non linear programming problem have been validated and compared on a set of 100 benchmark analytic test problems taken from literature.
The deterministic strategy BBWORHP is solving to optimality 93 problems over 100, with an average value of 22188 function evaluations. The violation of the constraints, for the required accuracy, is lower than 10^{-6}, while the error on the objective value ranges between -0.103 and 0.063, respectively for the CROP20 (number 98) and DU-OPT5 (number 63) problems. This means that the deterministic strategy was able to find a solution that improves the already known optimal value by 10.3%, for the test problem CROP20, and a solution that worsens it by 6.3% for the DU-OPT5 problem. Generally the BBWORHP algorithm was able to converge to a solution with an error lower than zero in the 26.9% of the solved problems while the rate of solutions with a relative error over zero is just of the 3.2%. All this results must be interpreted according to the tolerances used for optimality and feasibility in the NLP solver.
The 7 problems which BBWORHP was not able to solve present more than 20 integer variables and constraints. When solving integer programming problems using the branch and bound approach, the number of NLP sub-problems solved at each node of the decision tree grows exponentially with the number of integer variables involved. An upper limit of 1000 iterations is fixed for this test meaning a maximum of 2000 NLP sub-problems to be solved from the branching procedure that generates two relaxed optimization subproblems at each step. Therefore, a weaker restriction on the number of iterations can enable problems with a larger number of integer variables to reach convergence at the price of a higher computational load.
BBWORHP is a deterministic strategy with a global search capability in the discrete search space but it is just locally convergent with respect to the set of continuous variables. This makes the algorithm very sensitive to the choice of the initial guess. A good initial guess is provided in the implementation of every test problem and it has been used unchanged in all test cases. A different choice of the initial guess could improve the behavior of the algorithm. This is the case for example of the GEAR4 test problem that presents a small number of integer variables, hence its failure of convergence cannot be attributed to the computational load.
The stochastic strategy is able to solve to optimality (best feasible solution) 85 test problems over 100 with an average value of 254590 function evaluations. The error on the objective values ranges between -0.10 and 8.7 respectively for CROP20 (number 98) and NVS20 (number 36) problems. The algorithm was able to converge to a solution with an error lower than zero in 12.9% of the solved problems while the percentage of solutions returned that worsen the best known optimal value are 20%. Although the majority of the test problems (70.6%) are solved up to feasibility in the complete set of different runs with a standard deviation that in 36.5% of the cases is smaller than 0.1.

	$\overline{\text{feval}}$	pb solved	$\overline{\text{err}}$	err<0 (%)	err>0 (%)
BBWORHP	22188	93	0.0026	26.9	3.2
PSO	254590	85	0.2681	12.9	20

TABLE 5.5. Comparison of the results obtained with the two global strategies for single objective MINLP on the set of benchmark problems.

The issue in applying the stochastic strategy on the test problems set is the definition of the test problems themselves. Most of the unsolvable problems present large upper bounds for the real variables, in the order of 10^{10}. The definition of wide bounds enlarge the search space making the exploration difficult for the swarm. A better a priori knowledge of the optimization problem and its optimal solution could allow a redefinition of the problem narrowing the bounds on the optimization variables. This can potentially help the convergence of the stochastic strategy for the unsolved test problems. If such an additional information is not available, enlarging the size of the swarm is the only way to tackle the unsolved problems, but this will affect consistently the time required for optimization.
In light of the result presented, summarized in Table 5.5, the deterministic strategy based on the branch and bound algorithm for mixed integer non linear programming resulted more efficient than the stochastic one in terms of number of function evaluations to reach convergence and quality of the solution returned. $\overline{\text{feval}}$ and $\overline{\text{err}}$ note the mean values of the number of function evaluations and of the error from the best known optimal value. However, it has been experienced during the test phase that the performance of the BBWORHP algorithm is strictly related to the initial guess given to the optimizer and the scaling technique adopted. If a good initial guess is not available and the problem presents high multimodality, the stochastic strategy will turn out to be the best optimization approach even though it is computationally more expensive.

5.1.2. MDO Test Problem. The two single objective algorithms have been tested also on an applicative MDO problem. A complete MDO run is executed for the Ariane 5 ECA launcher, freezing all technological and architectural variables as well as the propulsion system design, and allowing a $\pm 30\%$ range for optimizable stages and boosters propellant masses, the length over diameter and trajectory loads. Moreover, to exploit the capabilities of the branch and bound algorithm to deal with categorical variables, the variable modeling the main structural material of the boosters is added to the optimization problem to demonstrate the validity of the problem reformulation presented in Section 3.4.1.2. The structural material can be choose from three options: aluminium (Al7075), steel or composite materials (CFRP), each identified by an integer value. The parameters related to the different materials that will influence the computation of the vehicle structural mass are density, minimum gage thickness, tensile yield and ultimate strengths and

(A) Set of vehicle design optimization variables.

Variable	id	LB	UB
LS propellant mass [tons]	$M_{\text{prop,EPC}}$	121.3	225.3
US propellant mass [tons]	$M_{\text{prop,ESC-A}}$	10.1	18.7
BS propellant mass [tons]	$M_{\text{prop,P241}}$	168.1	312.17
BS diameter [m]	D_{BS}	2.44	3.66
BS main struct. material [-]	SM_{BS}	{Al7075, Steel, CFRP}	

(B) Set of trajectory optimization variables.

Variable	id	LB	UB
Take off pitch deviation [deg]	$\Delta\theta_{\text{PO}}$	1	5
Take off pitch over duration [s]	Δt_{PO}	2	10
Take off pitch over decay time [s]	$\Delta t_{\text{PO,decay}}$	1	5
Take off pitch over heading angle [deg]	ψ_{PO}	-10	10
Atmospheric, pitch deviation [deg]	$\Delta\theta_{\text{EPC}}$	-10	20
Atmospheric, yaw deviation [deg]	$\Delta\psi_{\text{EPC}}$	-10	10
Exoatmospheric, yaw deviation [deg]	$\Delta\psi_{\text{ECA}}$	-10	10
BTL initial pitch [deg]	$\Delta\theta_{\text{BTL,i}}$	-50	50
BTL final pitch [deg]	$\Delta\theta_{\text{BTL,f}}$	0	50
BTL parameter [-]	ξ_{BTL}	-1	1

TABLE 5.6. Ariane 5 ECA MDO, minimization of the GTOW mass.

Young modulus (stiffness modulus).
The optimization objective is the minimization of the Gross Take-Off Weight (GTOW). The set of optimization variables and relative bounds is given in Table 5.6. Five variables are related to the launcher design while the remaining ones are trajectory control variables. The thrust level is fixed and the launcher is thrusting constantly at its maximum power.

The multidisciplinary optimization with the PSO algorithm is performed for 5 different runs with random initialization of the swarm. The default parameter setting is used as already presented in the analytic tests, and a swarm of 500 particles that explores the search space for a maximum number of 100 iterations is set. The number of function evaluations is hence 50000, corresponding to an average computational time of 12 hours.
The BBWORHP algorithm is run with optimality and feasibility tolerance of 10^{-4}, no information about the sparsity structure of the Jacobian and Hessian matrix is available. Hence finite difference and dense BFGS are used for the approximation of the first and second order derivatives. Two different initial guesses are given to the deterministic strategy: the default, corresponding to the mid point of the box constraints, to validate the global exploration capabilities of the method, and the best solution returned by the PSO to

(A) PSO results.

Variable	Actual	MDA	PSO best	PSO deviation
$M_{\text{prop,EPC}}$	173.3	173.3	145.5 (−16.0%)	11.6 (6.7%)
$M_{\text{prop,ESC-A}}$	14.4	14.4	17.3 (+20%)	2.1 (14.6%)
$M_{\text{prop,P241}}$	240.1	240.1	221.7 (−7.6%)	7.1 (2.9%)
D_{BS}	3.05	3.05	3.0 (−1.6%)	0.11 (3.6%)
SM_{BS}	Steel	Steel	CFRP	2 Steel, 3CFRP
GTOW [tons]	**766.4**	**770.8**	**691.1** (−9.8%)	**22.7** (3.0%)
CPU [h]	-	-	12	-

(B) BBWORHP results.

Variable	Actual	MDA	BBWORHP std	BBWORHP ref
$M_{\text{prop,EPC}}$	173.3	173.3	149.2 (-13.9%)	144.9 (-16.4%)
$M_{\text{prop,ESC-A}}$	14.4	14.4	12.4 (-13.9%)	17.0 (+18.0%)
$M_{\text{prop,P241}}$	240.1	240.1	228.7 (-4.7%)	221.7 (-7.7%)
D_{BS}	3.05	3.05	3.0 (-1.6%)	3.0 (-1.6%)
SM_{BS}	Steel	Steel	CFRP	CFRP
GTOW [tons]	**766.4**	**770.8**	**704.7** (-8.1%)	**690.0** (-10.0%)
CPU [h]	-	-	2.7	1.3

TABLE 5.7. Ariane 5 ECA "small" MDO, minimization of the GTOW mass.

investigate the efficiency of a local refinement with gradient based techniques close to a global optimal solution.
All optimization variables are scaled between 0 and 1 and objective and constraint functions are scaled with their reference values. In all results, design and trajectory constraints are satisfied below the allowed tolerances. In particular for the trajectory equality constraints an error of 50 km on the semi-axis, 0.002 on the eccentricity and 0.5 degrees on the inclination are allowed for reaching a Geostationary Transfer Orbit (GTO) with 24383.6 km of semi-axis, 0.7292 of eccentricity and 7 degrees of inclination.
The results are compared in Table 5.7 with the actual values of the vehicle design parameters and the value computed by the multidisciplinary analysis, to scale the contribution brought by the MDO in reducing the total mass with the approximation errors introduced by the model. The geometry of the best solutions found by the two algorithms are reported in Figure 5.1, in comparison with the design estimated by the MDA.

5.1.2.1. *Conclusions.* The two global optimization strategies, PSO and BBWORHP, have been compared on a single objective MDO problem with the purpose of redistributing the propellant mass between the existing Ariane 5 components, with a consequent effect on the geometry. The objective

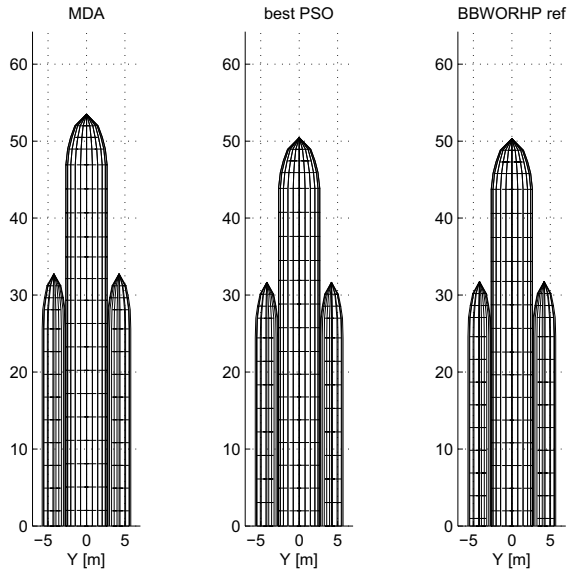


FIGURE 5.1. Best PSO and BBWORHP refinement solutions geometries. MDO problem of Ariane 5, minimization of the GTOW.

is the minimization of the total mass at launch for a mission to GTO with 10050 kg of payload. To test the capability of the deterministic approach to deal with categorical variables, the main structural material of the boosters is left optimizable between three values: aluminium, composites and steel. The best optimal solution found by the PSO and BBWORHP steers the design toward the use of composite materials that are lightweight. This is noticed in the decreasing of the inert masses of each booster from 33.3 tons of the MDA case (steel) to 25.3 tons of the optimized solution (composite materials). The resulting total mass of the rocket is reduced by 10% with respect to the actual mass. To assess the real MDO effect, one should consider also the overestimation due to the approximation models used inside the multidisciplinary analysis. In the Ariane 5 case the total mass at launch is overestimated by the MDA just by the 0.6%, hence the real MDO effect in reducing the GTOW is equal to 9.4%. The propellant in the boosters and in the lower stage is reduced respectively by the 16% and 8% while the upper stage is considerably increased, by 18%. The data reported here are related to the best solution found. It is the solution returned by the BBWORHP algorithm using the PSO solution as initial guess.

The BBWORHP algorithm is able to converge to an optimal solution of total mass at launch equal to 705 tons, 8% less than the actual one, starting with the initial guess equal to the actual design, in less than 3 hours. The solution found by the deterministic strategy with the standard initial guess is just a local optimum. The PSO algorithm is indeed capable of globally exploring the search space and converging to a better optimal solution. On the other

hand the stochastic process is much more inefficient ($\sim$ 12 hours of computational time) and the standard deviation on the final objective value between the different runs is of 22 tons. To reduce the standard deviation and assess the robustness of the stochastic process the number of iterations needs to be increased, hence affecting further the computational effort. The best solution returned by the PSO algorithm could be further improved by BBWORHP in slightly more than one hour, changing the final values of the continuous variables and leaving unchanged the optimal boosters structural material. This example clearly shows the local exploration capabilities of BBWORHP algorithm with respect to the continuous variables. The two solutions returned by the algorithm belong to different areas of the search space and regions of attraction of different local minima. The first one identifies a region where the minimization of the GTOW is accomplished reducing the propellant mass of each component. The second one describes a region of the search space, where an increase in the upper stage propellant mass can improve the vehicle performances at the cost of further reducing the masses in the lower stage and boosters components.
The deterministic strategy is the most efficient and effective strategy for the given problem. However, the applicability of the technique strongly depends on the number of discrete variables, the availability of a good initial guess or, if not available, the capability of reducing to the utmost the bounds on the continuous variables limiting the search space region.

5.2. Validation of Multiple Objective MINLP Techniques

The two global approaches for solving MINLP multi-objective problems presented in Section 3.4.2, the stochastic HGO algorithm and the deterministic MADS, are tested and compared on multi-objective analytic and MDO problems.

5.2.1. Analytic Test Problems. The selected multi-objective optimization strategies are validated on a set of unconstrained and constrained bi-objective optimization problems taken from literature [**146, 147**]. The bi-objective case, besides the simplicity of visualizing the corresponding Pareto Front, is sufficient to reflect the main issues of multi-objective optimization problems.
The quality of a multi-objective optimization algorithm is measured not only in term of convergence to the global optimal front but also on the ability of uniformly covering it, maintaining diversity between the solutions. The analytic test problems are chosen to tackle the main issues in accomplishing these two tasks. Concerning convergence, those are:

Multi-modality: as in the single objective case, multi-objective algorithms can get trapped in local Pareto frontiers.

Deception: the entire search space is attracted by a non-global optimum, the "deceptive attractor".

Isolated Optimum: the optimum is surrounded by unfeasible solutions or solutions that don't give any useful information to the algorithm to converge.

Furthermore, for maintaining the diversity of the solutions along the frontier, the following features might cause difficulties in the algorithms:

Convex or non-convex fronts: the fitness of a solution is generally assigned proportionally to the number of solutions it dominates, this makes it difficult to cover the extremes of the front and non convex Pareto frontiers.

Discontinuous fronts: the front is divided in disconnected regions. The comparison for dominance of different solutions belonging to different segments of the frontier can lead to the extinction of representative solutions in each of the disconnected areas.

No-uniform distribution: some regions in the front may have a higher density of solutions than others.

In [**148**] a method is proposed to construct a set of bi-objective test problems able to tackle all the features presented above. The general structure of the problem for the unconstrained case is

$$\min_{X \in \Omega \subseteq \mathbb{R}^{n_x}} (f_1(x_1), f_2(X)),$$

where $X = (x_1, ..., x_{n_x})$. The first objective $f_1 : \mathbb{R} \to \mathbb{R}$ is a function depending only on the first design variable, and the second objective $f_2 : \mathbb{R}^{n_x} \to \mathbb{R}$ is defined as

$$f_2(X) = g(x_2, ..., x_{n_x})h(f_1(x_1), g(x_2, ..., x_{n_x}))$$

with $g : \mathbb{R}^{n_x - 1} \to \mathbb{R}$ a function of the remaining $n_x - 1$ variables, and $h : \mathbb{R}^2 \to \mathbb{R}$ the composite function of g and the first objective. The advantage of this scheme is that the minimum of the function g corresponds to the global Pareto frontier, hence it can be computed analytically. Moreover every function introduced can recreate some of the main difficulties presented above. The function f_1 is controlling the search space along the optimal Pareto front, no-uniformity issues can be recreated with non-linear functions biasing the solutions density toward certain regions of the Pareto front. The function g is controlling the search space laterally, so it can reproduce the main convergence issues such as multimodality, deception and isolated optima. Finally, the function h varies the shape of the front, from convex to non-convex, continuous or disconnected, creating difficulties on the optimal Pareto coverage. $\Omega \subseteq \mathbb{R}^{n_x}$ is defined by the design variables box constraints. No hypothesis of continuity and differentiability is made for the presented functions as the selected algorithms don't require any a priori knowledge about their regularity.

The test functions differ in the choice of the functions g and h, the number of variables and bounds. Only one test problem involves discrete variables. The main purpose of the following tests is indeed to prove the convergence and coverage capabilities of the algorithms in challenging global optimization

problems, and not to prove that they are able to deal with discrete variables. This is already ensured by the nature of the algorithms.
Seven test functions tracing the structure presented above are introduced for the unconstrained case.

PROBLEM 5.2.1. The test function $\mathcal{T}_1$ has a **convex** optimal Pareto front. The optimization problem is defined as

$$\min_{X\in[0,1]^{30}}(f_1(x_1), f_2(X))$$

with

$$\begin{aligned} f_1(x_1) &= x_1, \\ g(x_2,...,x_{30}) &= 1+9\sum_{i=2}^{30}\frac{x_i}{29}, \\ h(f_1,g) &= 1-\sqrt{\frac{f_1}{g}}. \end{aligned}$$

The optimal Pareto front is formed with $g(x_2,...,x_{30})=1$.

PROBLEM 5.2.2. The test function $\mathcal{T}_2$ has a **non-convex** optimal Pareto front. The optimization problem is defined as

$$\min_{X\in[0,1]^{30}}(f_1(x_1), f_2(X))$$

with

$$\begin{aligned} f_1(x_1) &= x_1, \\ g(x_2,...,x_{30}) &= 1+9\sum_{i=2}^{30}\frac{x_i}{29}, \\ h(f_1,g) &= 1-\left(\frac{f_1}{g}\right)^2. \end{aligned}$$

The optimal Pareto front is formed with $g(x_2,...,x_{30})=1$.

PROBLEM 5.2.3. The test function $\mathcal{T}_3$ has a convex local and global optimal fronts with a very narrow basin of attraction of the global front reproducing the **isolated optimum** problem case. The optimization problem is defined as

$$\min_{X\in[0.1,1]\times[0,1]}(f_1(x_1), f_2(X))$$

with

$$\begin{aligned} f_1(x_1) &= x_1, \\ g(x_2) &= 2-e^{-(\frac{x_2-0.2}{0.004})^2}-0.8e^{-(\frac{x_2-0.6}{0.4})^2}, \\ h(f_1,g) &= g(x_2)/x_1. \end{aligned}$$

The global optimal Pareto front is formed with $g(x_2)\approx 0.7057$ while the local one with $g(x_2)=1.2$.

PROBLEM 5.2.4. The test function $\mathcal{T}_4$ has a **disconnected** optimal Pareto **front**. The optimization problem is defined as

$$\min_{X\in[0,1]^{30}}(f_1(x_1), f_2(X))$$

with

$$\begin{aligned} f_1(x_1) &= x_1, \\ g(x_2,...,x_{30}) &= 1 + 9\sum_{i=2}^{30}\frac{x_i}{29}, \\ h(f_1,g) &= 1 - \sqrt{\frac{f_1}{g}} - \left(\frac{f_1}{g}\right)\sin(10\pi f_1). \end{aligned}$$

The optimal Pareto front is formed with $g(x_2,...,x_{30}) = 1$. The introduction of the trigonometric function in the h function causes discontinuity in the optimal Pareto front that is formed by a set of disconnected convex parts. The discontinuity in the front doesn't affect the continuity in the design variables space.

PROBLEM 5.2.5. The test function $\mathcal{T}_5$ has 21^9 local optimal Pareto fronts, it is a **multimodal** optimization problem. The optimization problem is defined as

$$\min_{X\in[0,1]\times[-5,5]^9}(f_1(x_1), f_2(X))$$

with

$$\begin{aligned} f_1(x_1) &= x_1, \\ g(x_2,...,x_{10}) &= 1 + 90 + \sum_{i=2}^{10}(x_i^2 - 10\cos(4\pi x_i)), \\ h(f_1,g) &= 1 - \sqrt{\frac{f_1}{g}}. \end{aligned}$$

The global optimal Pareto front is formed with $g(x_2,...,x_{10}) = 1$, the best local optimal Pareto front with $g(x_2,...,x_{10}) = 1.25$

PROBLEM 5.2.6. The test function $\mathcal{T}_6$ includes two difficulties caused by the **non uniformity** of the search space: the optimal Pareto solutions are not uniformly distributed along the global front and the density of the solutions is lower close to the front and higher away from it

$$\min_{X\in[0,1]^{10}}(f_1(x_1), f_2(X))$$

with

$$\begin{aligned} f_1(x_1) &= 1 - \exp(-4x_1)\sin^6(6\pi x_1), \\ g(x_2,...,x_{10}) &= 1 + 9\left(\frac{\sum_{i=2}^{10}x_i}{9}\right)^{1/4}, \\ h(f_1,g) &= 1 - \left(\frac{f_1}{g}\right)^2. \end{aligned}$$

The optimal Pareto front is formed with $g(x_2, ..., x_{10}) = 1$ and is nonconvex.

PROBLEM 5.2.7. The test function $\mathcal{T}_7$ describe a **deceptive** problem. x_1 represents a binary string. The optimization problem is defined as

$$\min_{X \in \{0,1\}^{30} \times (\{0,1\}^5)^{10}} (f_1(x_1), f_2(X))$$

with

$$\begin{aligned} f_1(x_1) &= 1 + u(x_1), \\ g(x_2, ..., x_{11}) &= \sum_{i=2}^{11} v(u(x_i)), \\ h(f_1, g) &= \frac{1}{f_1}. \end{aligned}$$

where $u(x_i)$ is the function returning the number of ones in the bit vector x_i and

$$v(u(x_i)) = \begin{cases} 2 + u(x_i) & \text{if } u(x_i) < 5 \\ 1 & \text{if } u(x_i) = 5. \end{cases}$$

The true optimal front is formed with $g(x_2, ..., x_{11}) = 10$ while the deceptive attractive front is with $g(x_2, ..., x_{11}) = 11$. All fronts are convex.

Constraints are added to the first test problem $\mathcal{T}_1$ for generating the set of constrained test problems. Some difficulties arise when introducing constraints, in particular difficulties in the vicinity of the optimal front due to the particular shape of the feasible region, and in the entire search space due to the presence of disconnected feasible areas. Seven test problems are considered with a generic tunable constraint that reproduces the most interesting cases.

PROBLEM 5.2.8. The test function $\mathcal{T}_8$ has two non linear inequality constraints

$$c_i(X) \equiv f_2(X) - a_i \exp(-b_i f_1(x_1)) \geq 0, \quad i = 1, 2$$

with $a_1 = 0.858$, $a_2 = 0.728$ and $b_1 = 0.541$, $b_2 = 0.295$; $f_1(x_1)$ and $f_2(X)$ are defined as in $\mathcal{T}_1$ and $X \in [0, 1]^{30}$.

The optimal Pareto front is formed by a portion of the convex unconstrained frontier of $\mathcal{T}_1$ and two joined segments coming from the bounds of the feasible region defined by constraints c_1 and c_2, as shown in Figure 5.2. This means that in the vicinity of the optimal front the two non linear constraints are active. The difficulties in the vicinity of the front lie in its edges coming from the intersection of the constraints graphs. The complexity of the first test problem can be increased augmenting the number of constraints and hence the number of intersecting segments or introducing the same constraints to a multimodal or deceptive unconstrained problem.

PROBLEM 5.2.9. The test functions $\mathcal{T}_9$ to $\mathcal{T}_{14}$ have a single non linear inequality constraint $c(X) \geq 0$ that depends on five parameters

$$\begin{aligned} c(X) = \; & \cos(\theta)(f_2(x) - e) - \sin(\theta) f_1(x) - \\ & + a |\sin(b\pi(\sin(\theta)(f_2(x) - e) + \cos(\theta) f_1(x))^c)|^d \end{aligned}$$

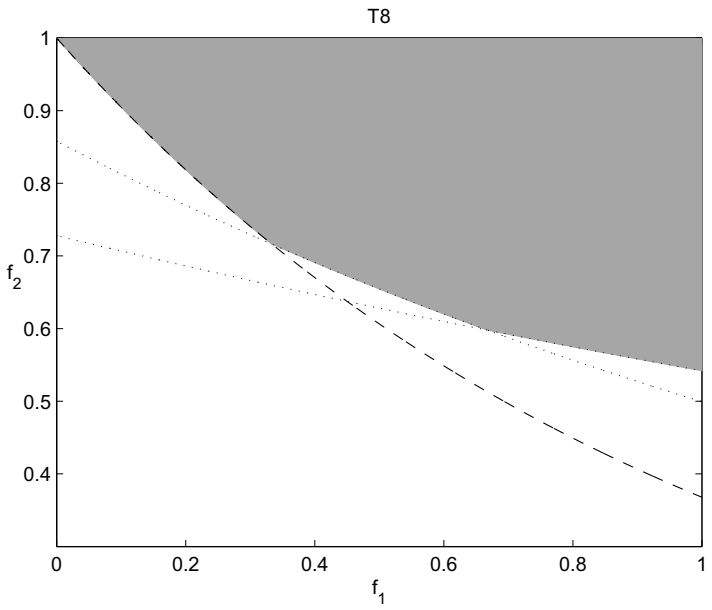


FIGURE 5.2. Feasible region in the objective space for problem test $\mathcal{T}_8$.

where $f_1(x_1)$ and $f_2(X)$ are defined as in $\mathcal{T}_1$ and the set of parameters $(\theta, a, b, c, d, e) \in \mathbb{R}^5$ are defined for each problem as

$$\begin{aligned}
\mathcal{T}_9 &\Rightarrow (-0.2\pi, 0.2, 10, 1, 6, 1),\\
\mathcal{T}_{10} &\Rightarrow (-0.2\pi, 0.1, 10, 1, 0.5, 1),\\
\mathcal{T}_{11} &\Rightarrow (-0.2\pi, 0.75, 10, 1, 0.5, 1),\\
\mathcal{T}_{12} &\Rightarrow (-0.2\pi, 0.1, 10, 2, 0.5, 1),\\
\mathcal{T}_{13} &\Rightarrow (0.1\pi, 40, 0.5, 1, 2, -2),\\
\mathcal{T}_{14} &\Rightarrow (-0.05\pi, 40, 5, 1, 6, 0).
\end{aligned}$$

The unconstrained Pareto front for these test cases is completely infeasible. The first four test problems reflect difficulties in the **vicinity** of the Pareto front that is formed by isolated discrete points. The challenge for the algorithms is to converge and maintain diversity between the solutions in disconnected regions. The remaining two test problems have difficulties in the **entire** search space, that is formed by disconnected feasible regions, in approaching the optimal front longitudinally and vertically as shown in Figure 5.3. The unconstrained front is plotted as a dashed line while the dotted line is the locus of points where the constrained front lies on

$$(f_2(X) - e)\cos(\theta) = f_1(x_1)\sin(\theta).$$

Each parameter has an influence on the form, shape and position of the frontier. For the first four cases the locus of points is controlled by the parameters e and θ. e is the intercept on the y-axis while $-e/\tan(\theta)$ is the intercept on the x-axis, so the two parameters are controlling the position of the front in

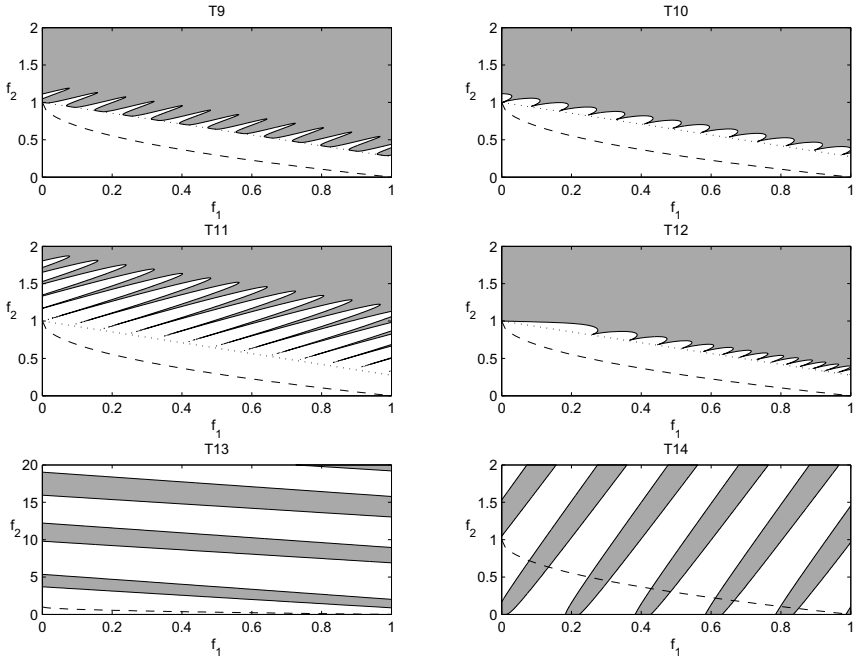


FIGURE 5.3. Feasibility study for test problems $\mathcal{T}_9$ to $\mathcal{T}_{14}$

the search space. The other parameters control the shape and the complexity of the derived front: b controls the number of disconnected regions, d the width of such regions (small values of d leads to single point solutions in each disconnected part), a controls the length of the corridors of passage between feasible and unfeasible regions in the search space (higher values of a make the convergence to the front more difficult), c controls the distribution of the disconnected parts of the front, for value of $c \neq 1$ the regions are not uniformly distributed. The latter two test cases are obtained from a complete different set of parameters, θ controls the slope of the feasible regions, moving the main difficulties, from the vicinity to the front, to the entire search space. As before, e is controlling the position of the disconnected regions in the objectives plane, d their width and b their quantities. The higher value of a makes the feasible region disconnected and c equally distributed as before.

The multi-objective global MINLP algorithms have been tested on the complete set of optimization problems. For the stochastic strategies a set of parameters needs to be tuned. Their choice is often critical for the success or failure of the algorithm. Due to the stochastic nature of the search methods and the variety of test problems, a globally valid set of parameters is far from being ever defined. The parameters of the evolutionary strategies selected for the comparison study and used in the developed hybridization techniques are fixed to a set of values initially suggested in the reference papers and then

DGMOPSO parameters	
Outher depth	4
Inner depth	1
Error threshold for the grid reset	0.03
Inertia parameter (linear)	[0.5, 0.2]
Self confidence (constant)	1.0
Swarm confidence (constant)	1.7
Mutation probability (constant)	0.1
Mutation distribution (linear)	[0, 5]
NSGA2 parameters	
Crossover probability	0.8
Crossover distribution	20
Mutation probability	$1/m$
Mutation distribution	10
MOACOr parameters	
Speed of convergence ρ	0.85
Locality of search q	0.0001
Number mutated sols	$n/10$
HGO parameters	
Threshold rate	0.1
Number of sub-iteration	10
Hybridization coefficient h_1	1/3
Hybridization coefficient h_2	1/3

TABLE 5.8. Default parameters for Multi objectives stochastic strategies.

further tuned with dedicated analyses described in [**118**]. These are reported in Table 5.8 and also refer to the default values taken for each algorithm in the hybrid strategy.
The number of solutions in the initial set is 100 and 300 is the maximum number of iterations, in such a way that the total number of function evaluations is equal to 30000, that is the maximum number of black box evaluations set for the MADS algorithm. The algorithms are compared in terms of number of function evaluations. Furthermore, also the results related to the single evolutionary strategies employed in the HGO algorithm are discussed to remark on the advantage of a hybrid approach. The comparison between the convergence capability of the stochastic techniques are analyzed with the MDR criteria with an accuracy of $\varepsilon = 10^{-5}$ on the domination rate. This means that the solution a is considered dominated by the solution b if

$$b \preceq (a - \varepsilon).$$

This allows to consider as improved solution, one which contributes to reduce at least one of the objectives with a deviation from the current non dominated solution larger than ε.
The consolidation ratio criteria is not giving useful information in this setting: to evaluate the spread of the solution along the front the number of solutions in the final archive is kept small. This is resulting in a small value for the CR operator in most of the test cases because in each iteration, even if convergence is reached, the solutions continue to mutate to spread along the frontier.
To reduce the stochastic effect on the evolutionary algorithms 10 runs for each problem have been executed. The results reported here are referring to the best run obtained in terms of convergence, estimated as the run with the final smallest value of the MDR operator. The CPU times reported is the average time required for each run.
Since the MDR criterion compares two successive fronts and the hybrid strategy recombines the front given by the three algorithms at every super-iteration, an external routine for computing the recombination of the archive at every sub-iteration has been implemented just for testing purpose since it affects consistently the algorithm efficiency.
The MADS algorithm returns the maximum value of dominated points found in the given number of black box evaluations. There is no operator for the evaluation of the spread of the solutions along the front, hence the comparison with the stochastic strategies is just in terms of convergence to the global optimal front and computational efficiency.

5.2.1.1. *Unconstrained optimization test problems.* The results obtained with the different optimization strategies for the unconstrained test problems are reported in Table 5.9. The best values attained for the MDR operator are marked in red, whereas the smallest CPU time required for optimization is shown in green. As a general consideration, DGMOPSO algorithm is the computationally most efficient algorithm while the HGO algorithm is the one with the best convergence rate (smaller values of the MDR operator) in the majority of the problems. Each problem is singularly analyzed hereafter and the plots of the obtained Pareto frontiers and convergence histories are reported. The overlapping of the solution sets of the different algorithms over the analytic frontier complicates their distinction. The spread of solutions along the fronts are hence visualized through the projection of the front on the abscissa. Please notice that the analytic Pareto front is plotted as a continuous line also in case of disconnected fronts, for which obviously only the not dominated segments, or set of points, for the optimal front are shown.

The first optimization problem, $\mathcal{T}_1$ doesn't present important difficulties. All the algorithms are able to converge to the optimal front maintaining a good spread of the solutions as shown in Figure 5.4. The hybrid algorithm has the best rate of convergence for the MDR criterion while DGMOPSO the largest

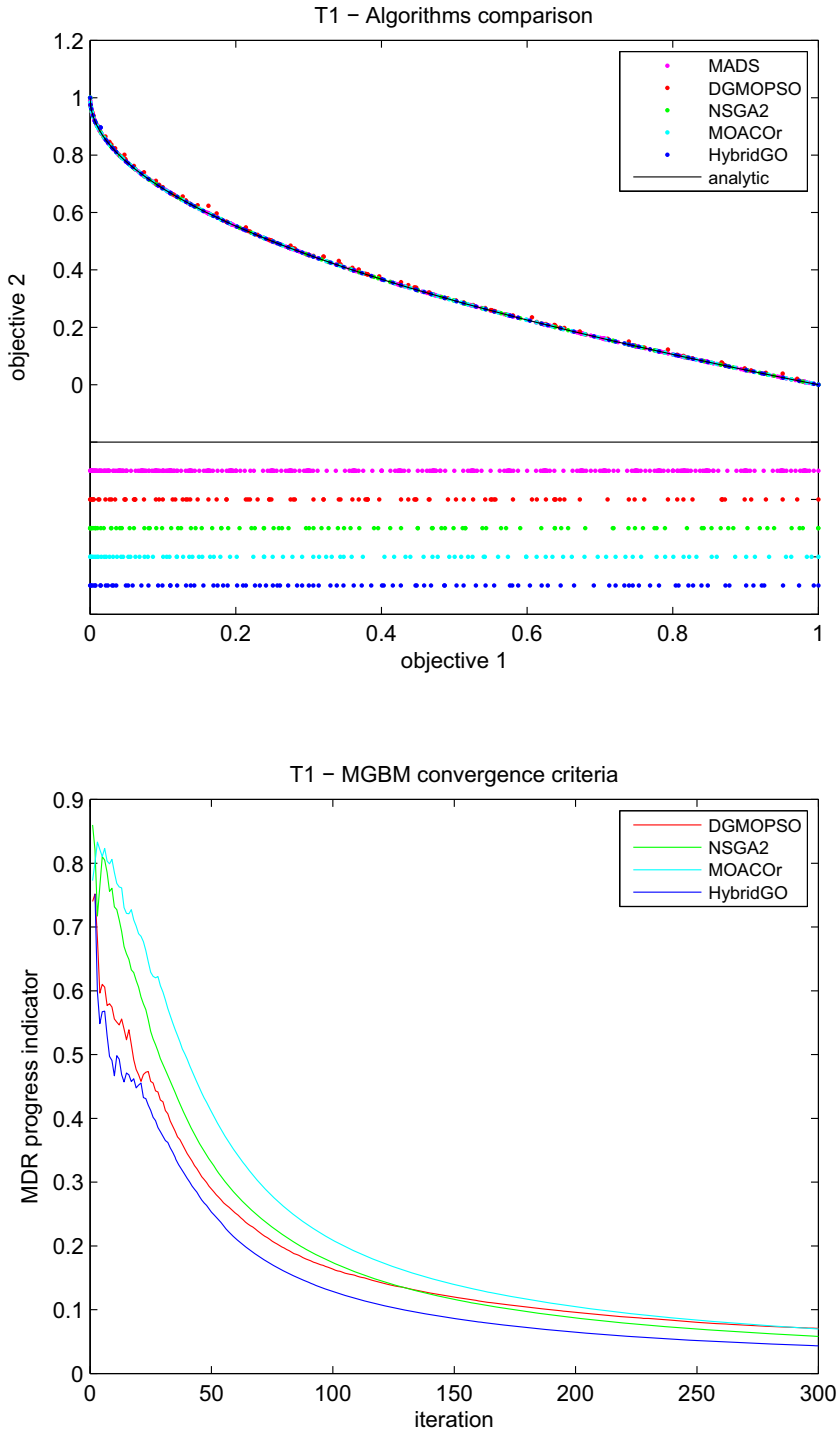


FIGURE 5.4. Pareto frontier and convergence comparison, using the MGBM criterion, of evolutionary strategies and MADS algorithms for the unconstrained multi objective optimization test problem $\mathcal{T}_1$.

		DGMOPSO	NSGA2	MOACOr	HGO	MADS
$\mathcal{T}_1$	CPU [s]	2.26	2.68	3.38	6.09	11.05
	I_{mdr}	0.0707	0.0581	0.0698	0.0433	-
$\mathcal{T}_2$	CPU [s]	2.08	2.53	3.29	5.84	7.88
	I_{mdr}	0.0256	0.0456	0.0432	0.0168	-
$\mathcal{T}_3$	CPU [s]	0.99	1.18	1.52	2.53	112.81
	I_{mdr}	0.0300	0.0146	0.0029	0.0076	-
$\mathcal{T}_4$	CPU [s]	2.18	2.68	3.21	5.38	4.72
	I_{mdr}	0.0600	0.0477	0.0630	0.0407	-
$\mathcal{T}_5$	CPU [s]	0.52	1.34	1.69	2.24	1.92
	I_{mdr}	0.0561	0.0791	0.0912	0.0802	-
$\mathcal{T}_6$	CPU [s]	1.31	1.48	1.14	2.82	2.54
	I_{mdr}	0.0154	0.0981	0.0145	0.0110	-
$\mathcal{T}_7$	CPU [s]	1.01	5.48	5.03	10.44	3.98
	I_{mdr}	0.0063	0.0093	0.0061	0.0071	-

TABLE 5.9. CPU and convergence comparison for the unconstrained test problem set.

one. This is reflected also in the respective Pareto frontiers. The solutions of the particle swarm algorithm are not covering completely the analytic front. Additional iterations are needed by the DGMOPSO algorithm to improve the current non dominated set of solutions. On the other hand DGMOPSO has the best efficiency rate. The allocated computational resources for DGMOPSO are comparable to the one of the NSGA2 and MOACOr but largely better than the one required by the hybrid and the deterministic strategies to perform the same number of function evaluations. The expensive computational load of HGO is attributable to the expensive internal operations needed for recombining and reordering of the external archive at every main iteration. The efficiency of the MADS algorithm is instead strictly related to the dimension of the current solution set that enlarges at every iteration. Moreover the technique implemented in MADS for solving bi-objective optimization problems building up a sequence of single objective problems, as expected, largely affect the computational efficiency of the method as it will be noticed in all the following test cases.

The second optimization problem, $\mathcal{T}_2$, represents the non-convex counterpart of the previous one. As before no main difficulties are identified, all the algorithms are able to converge to the global optimal front, most of them maintaining a good spread of the solutions as shown in Figure 5.5. The capability of an algorithm, of spreading the solutions along the frontier, is related to the shape of the Pareto front. The tails, and the regions of the

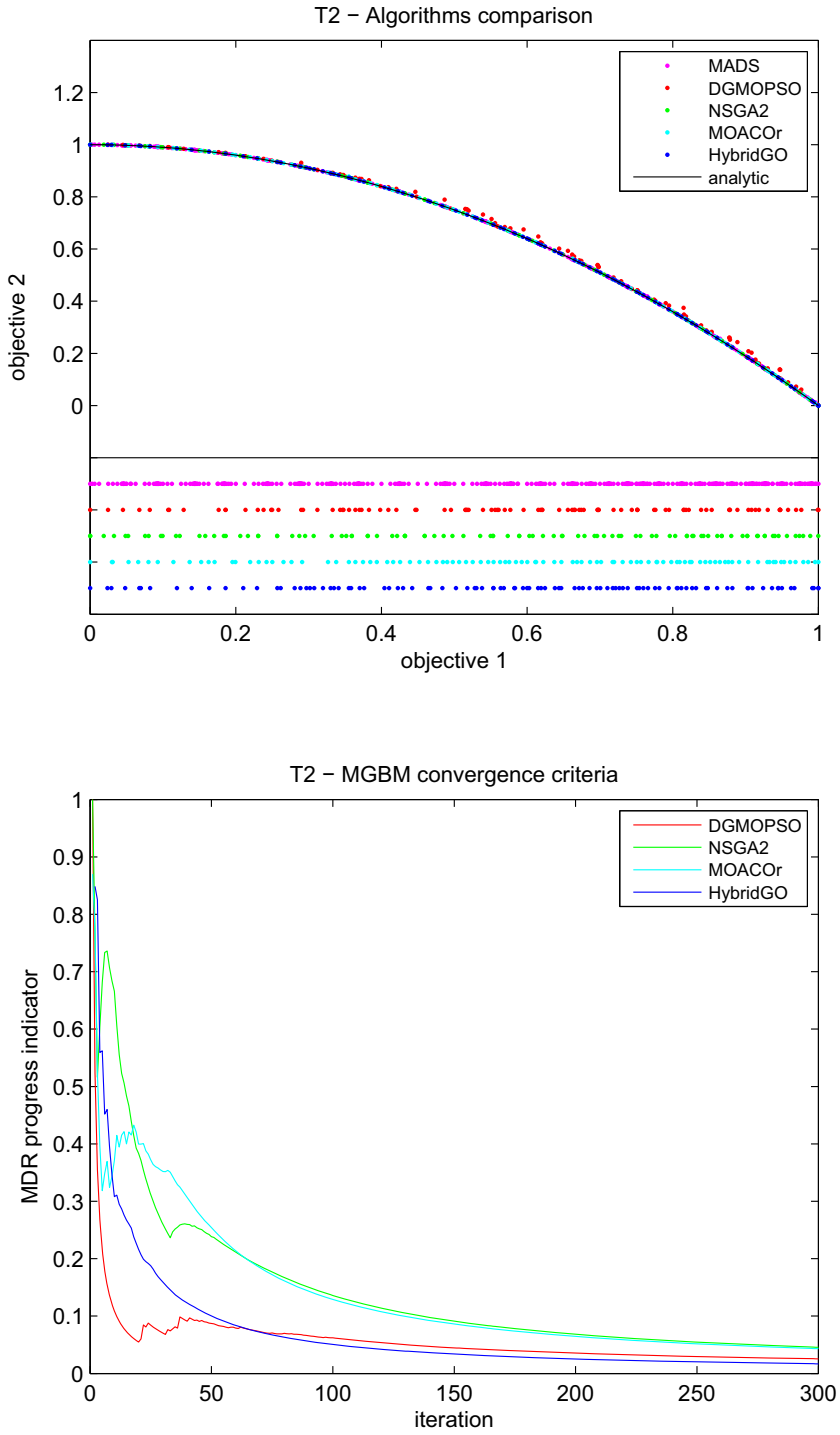


FIGURE 5.5. Pareto frontier and convergence comparison, using the MGBM criterion, of evolutionary strategies and MADS algorithms for the unconstrained multi objective optimization test problem $\mathcal{T}_2$.

frontier where a deviation from the current position records smaller deviation in one of the two objectives, are the most difficult areas to detect new non dominated solutions. This is because a solution, belonging to the tails, dominates a region of the search space smaller than the one dominated by a solution in a central position while two solutions with small deviation in one of the objectives dominate nearly the same area. The comparison in terms of computational efficiency and convergence rate remains unvaried from the previous case with the difference that the particle swarm algorithm achieves smaller values of the MDR operator still maintaining a subset of solutions dominated by all the other strategies. This means that further iterations cannot guarantee improvements in its current solution set.

The third benchmark problem, $\mathcal{T}_3$ presents two convex fronts, the global one corresponding to the isolated optimum of the function g and a local one. All algorithms, except MOACOr are able to converge to the global front as shown in Figure 5.6. The ant colony algorithm reaches convergence remaining trapped in the local frontier. The fastest rate of convergence to the global front is then the one of the hybrid strategy. The ratio between different computational times is consistent with the previous results. The average computational time is smaller due to the low number of optimization variables characterizing the problem. The large time required by the MADS algorithm is attributable to the dense solution set which covers entirely the analytic frontier with more than 15000 points as shown in the projection of Figure 5.6.

The fourth problem, $\mathcal{T}_4$ is characterized by a disconnected Pareto front, which represents its main difficulty. All algorithms are able to cover the whole disconnected regions of the front with a good spread of the solutions as shown in Figure 5.7. As in the first two test cases the DGMOPSO algorithm needs additional iterations to converge entirely to the analytic frontier as reflected by the corresponding value of the MDR operator that is the highest obtained between the stochastic algorithms. However it remains to be the computationally most efficient strategy.

The fifth benchmark problem, $\mathcal{T}_5$ is much more challenging than the previous ones. The main difficulty lays in the high number of local minima that contribute in deceiving the optimization algorithms in many local basins of attractions. The NSGA2, HGO and MADS algorithms are outperforming the remaining optimization strategies. This particular case puts in light the intrinsic behavior of the hybrid technique which promotes the genetic algorithm in converging to the optimal front as shown in Figure 5.8. DGMOPSO and MOACOr are not able to get close to the optimal front and no further improvements can be expected for the small values attained by the MDR operator. The computational times are reduced by the small number of optimization variables.

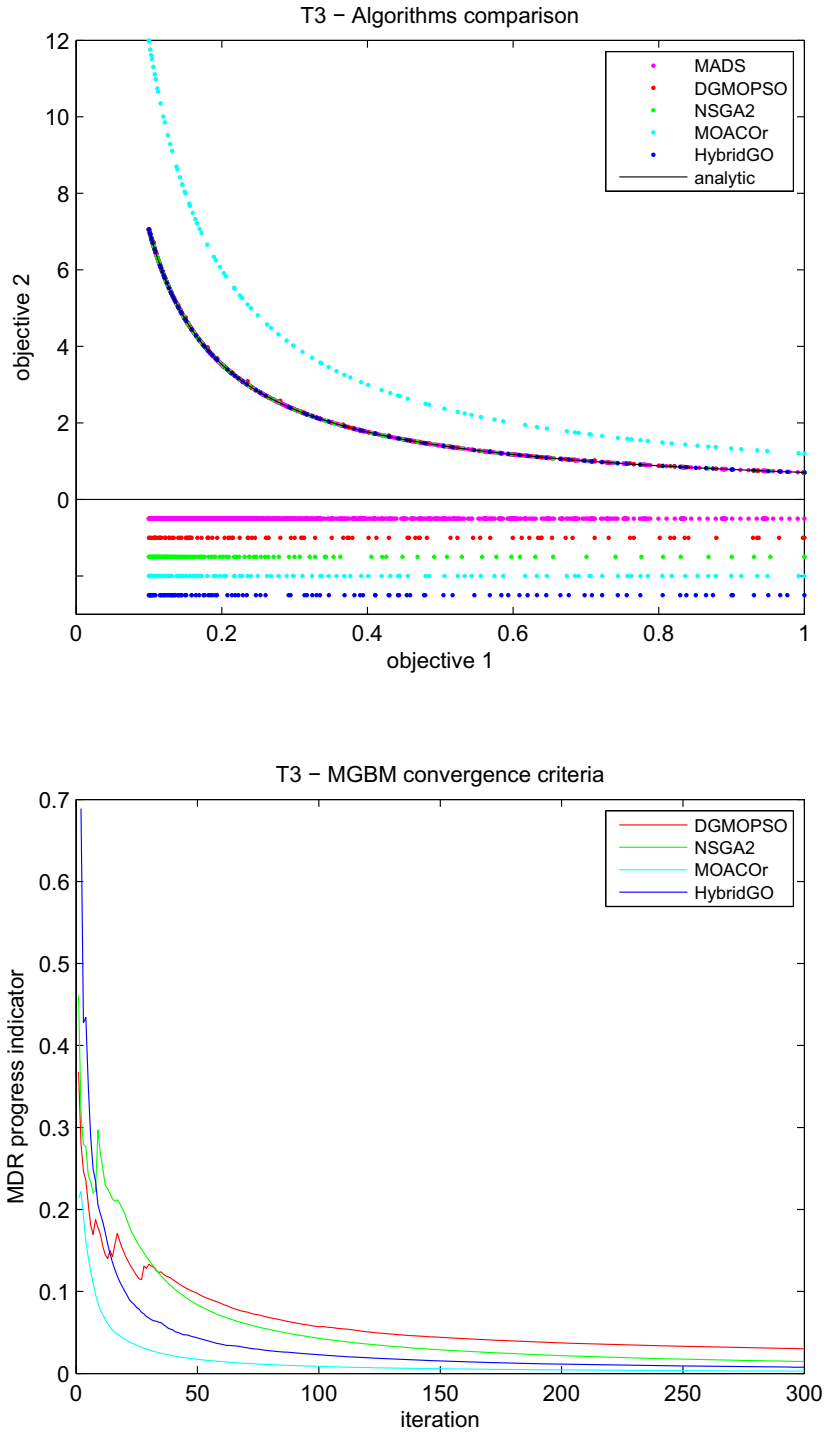


FIGURE 5.6. Pareto frontier and convergence comparison, using the MGBM criterion, of evolutionary strategies and MADS algorithms for the unconstrained multi objective optimization test problem $\mathcal{T}_3$.

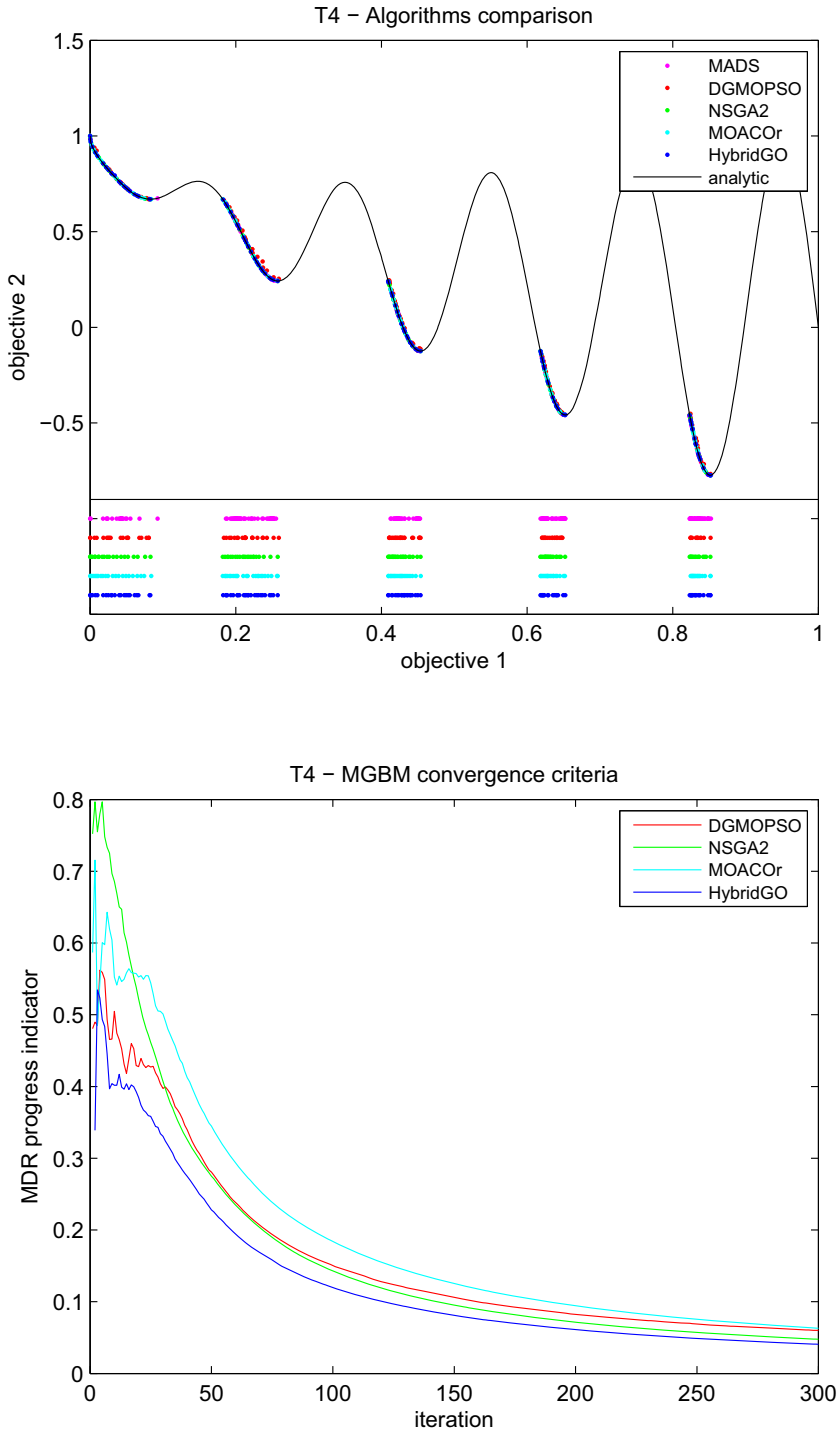


FIGURE 5.7. Pareto frontier and convergence comparison, using the MGBM criterion, of evolutionary strategies and MADS algorithms for the unconstrained multi objective optimization test problem $\mathcal{T}_4$.

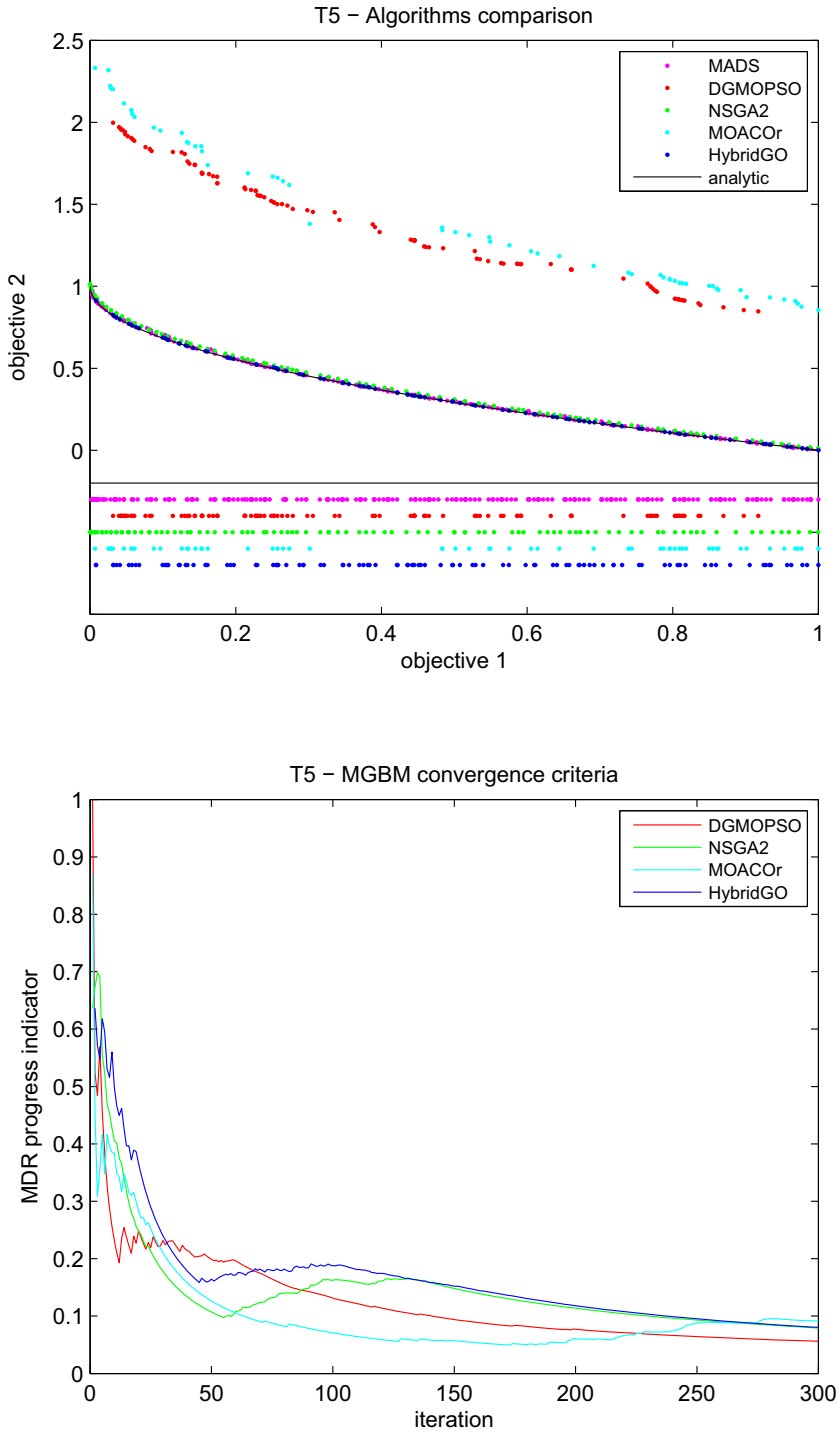


FIGURE 5.8. Pareto frontier and convergence comparison, using the MGBM criterion, of evolutionary strategies and MADS algorithms for the unconstrained multi objective optimization test problem $\mathcal{T}_5$.

The sixth benchmark problem $\mathcal{T}_6$, tests the ability of the algorithms of dealing with highly biased search spaces. The global optimal front is in a region of the search space with a low density of solutions. The NSGA2 strategy is not able to converge to the global optimal front while the MADS algorithm is not able to uniformly spread the solutions along the frontier as shown in Figure 5.9. DGMOPSO, MOACOr and HGO are able to converge to the optimal front with comparable convergence rate. The high value of the MDR operator, for the NSGA2 case, envisages the convergence of the algorithm to the true frontier in a higher number of iterations.

The last unconstrained test problem, $\mathcal{T}_7$ is a binary deceptive optimization problem. It has a discrete optimal front with the solutions equally spaced along the abscissa. The first objective is determined by the 30 bit unitation of the first variables plus one, thus f_1 can take values from 1 to 31 that is the maximum number of solutions existing in the optimal front. Only the hybrid strategy and the MADS algorithm, are able to converge to the complete solution set as shown in Figure 5.10. The solutions of the deterministic strategy are further dominating the ones of the hybrid technique, stating it as the most effective approach for the given problem. The values attained for the MDR operator by the different stochastic strategies are comparable and, as in the most of the presented test cases, the DGMOPSO algorithm is the most efficient technique.

5.2.1.2. *Constrained optimization test problems.* The results obtained with the different optimization strategies for the constrained test problems are reported in Table 5.10. As for the unconstrained case, in red are marked the best values attained for the MDR operator and in green the smallest CPU time required for optimization. Also in this case the DGMOPSO algorithm is generally the most computationally efficient strategy and the hybrid approach is the one with the highest convergence rate in almost the entire set of test problems. Each problem is singularly analyzed hereafter.

The first constrained optimization problem, $\mathcal{T}_8$ presents difficulties in the vicinity of the Pareto front. The optimal frontier is composed by part of the $\mathcal{T}_1$ unconstrained front and by the intersection of the boundary of the two feasible regions defined by the two inequality constraints. All the multi-objective algorithms are able to converge to the optimal front experiencing, as expected, the main difficulties in spreading the solutions along the constraints bounds as shown in Figure 5.11. In particular the NSGA2 algorithm is not able to cover the bound defined by the second constraint on the right tail of the frontier. Also the MADS algorithm experiences coverage problems in the same extreme part of the front. DGMOPSO, MOACOr and HGO show the best behavior, the latter one achieving convergence with the smallest value of the MDR operator. In terms of computational time, as for the unconstrained case, DGMOPSO is the most efficient algorithm while the hybrid algorithm

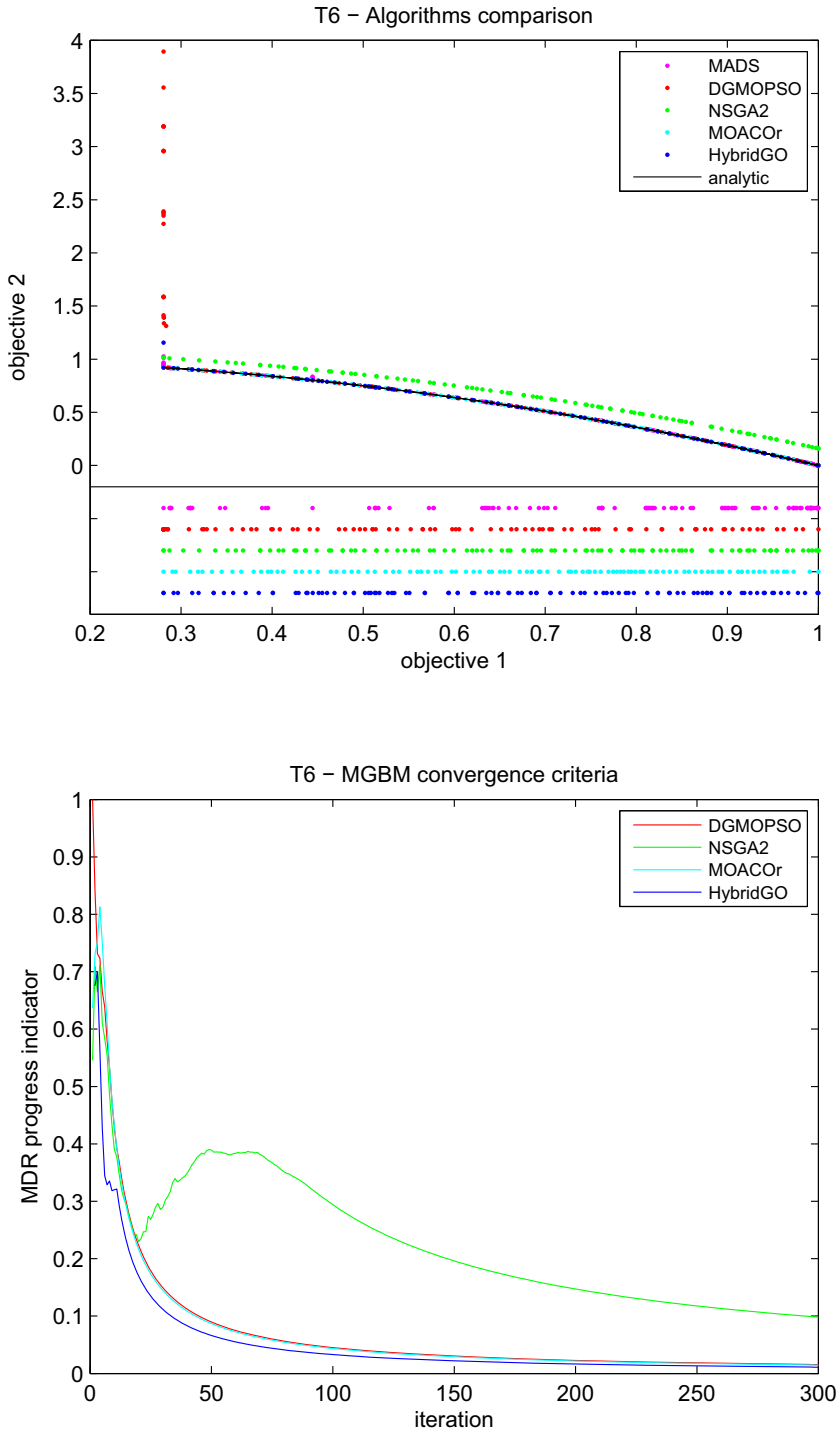


FIGURE 5.9. Pareto frontier and convergence comparison, using the MGBM criterion, of evolutionary strategies and MADS algorithms for the unconstrained multi objective optimization test problem $\mathcal{T}_6$.

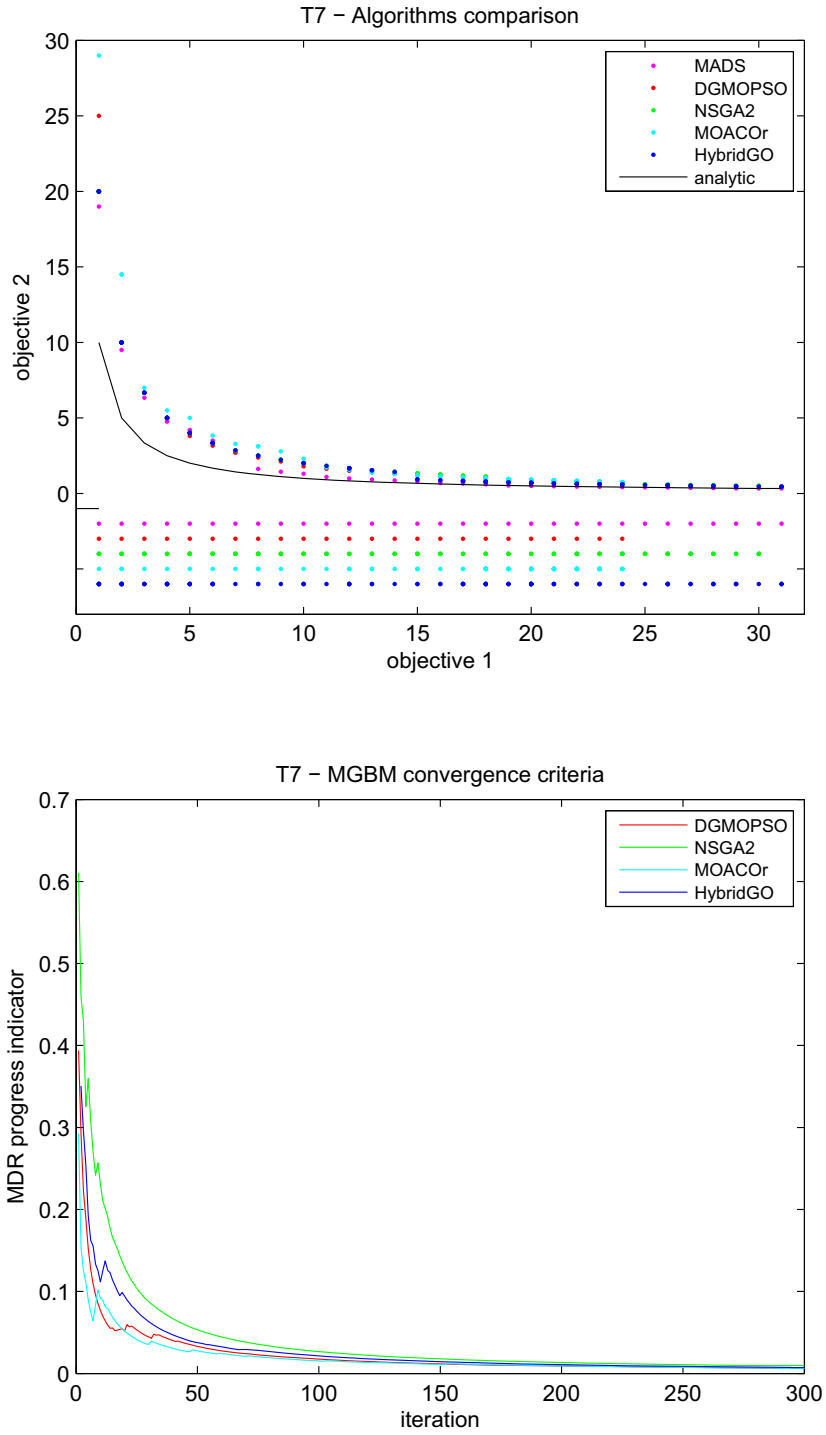


FIGURE 5.10. Pareto frontier and convergence comparison, using the MGBM criterion, of evolutionary strategies and MADS algorithms for the unconstrained multi objective optimization test problem $\mathcal{T}_7$.

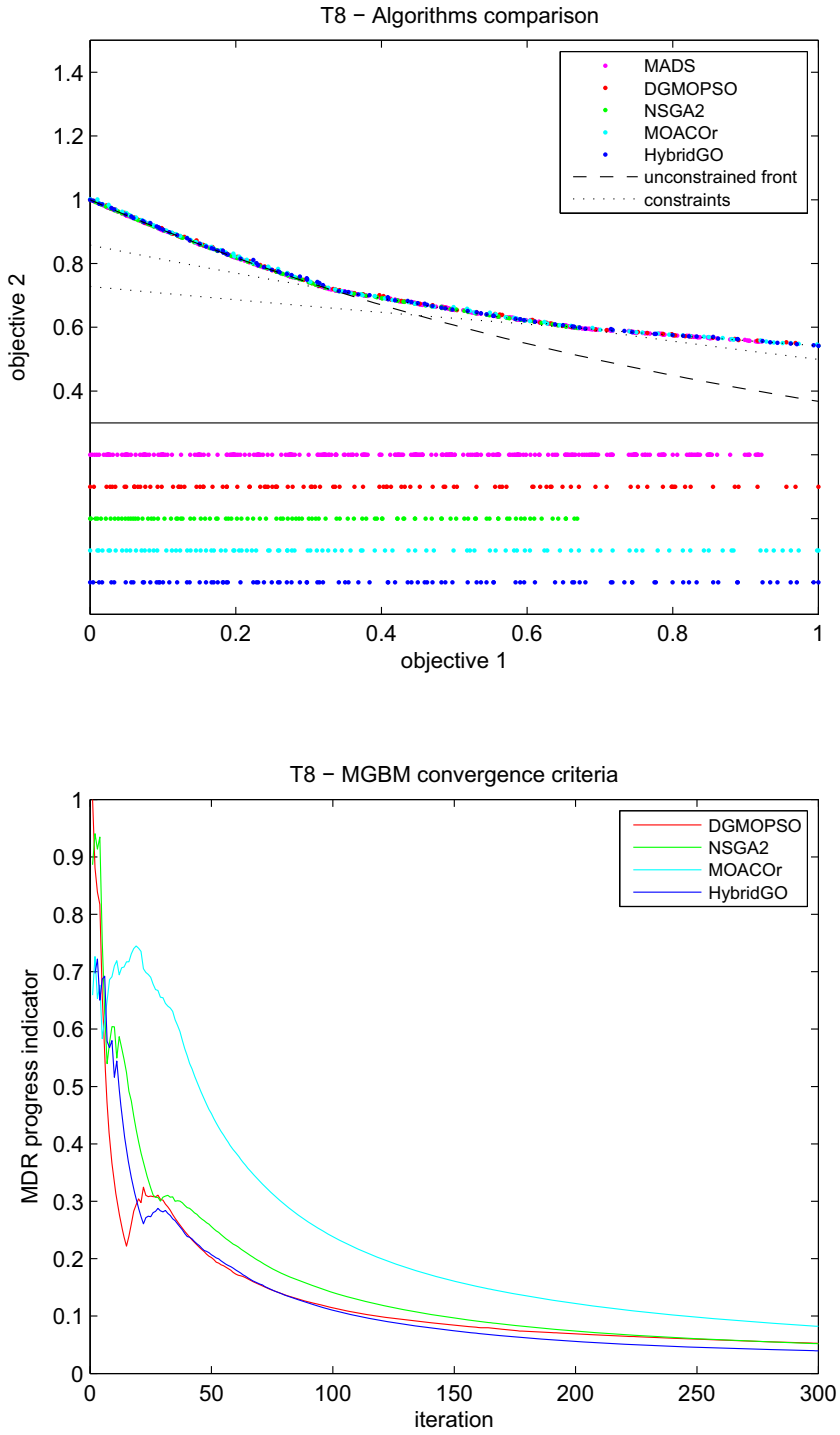


FIGURE 5.11. Pareto frontier and convergence comparison, using the MGBM criterion, of evolutionary strategies and MADS algorithms for the unconstrained multi objective optimization test problem $\mathcal{T}_8$.

		DGMOPSO	NSGA2	MOACOr	HGO	MADS
$\mathcal{T}_8$	CPU [s]	2.27	2.75	3.38	4.37	5.47
	I_{mdr}	0.0525	0.0518	0.0821	0.0393	-
$\mathcal{T}_9$	CPU [s]	1.78	2.63	2.98	3.61	23.24
	I_{mdr}	0.0521	0.0415	0.0420	0.0388	-
$\mathcal{T}_{10}$	CPU [s]	0.64	2.10	1.49	3.37	29.16
	I_{mdr}	0.0482	0.0591	0.0458	0.0366	-
$\mathcal{T}_{11}$	CPU [s]	0.43	1.81	1.49	3.11	16.34
	I_{mdr}	0.0450	0.0354	0.0354	0.0291	-
$\mathcal{T}_{12}$	CPU [s]	1.31	2.62	2.33	3.51	23.87
	I_{mdr}	0.0485	0.0455	0.0405	0.0393	-
$\mathcal{T}_{13}$	CPU [s]	2.34	2.60	3.71	3.92	2.49
	I_{mdr}	0.0204	0.0103	0.0117	0.0129	-
$\mathcal{T}_{14}$	CPU [s]	1.73	2.57	3.10	4.51	2.16
	I_{mdr}	0.0760	0.0877	0.0785	0.0635	-

TABLE 5.10. CPU and convergence comparison for the constrained test problem set.

and the deterministic strategy are the most inefficient ones.

The second constrained optimization problem, $\mathcal{T}_9$ tests the ability of the algorithms in covering as many as possible disconnected regions of the optimal front with a homogeneous spread of solutions in each segment. While all the evolutionary strategies are able to entirely cover the Pareto front with a good spread between the disconnected regions and a comparable convergence rate, the deterministic technique experiences problems in the vicinity of the Pareto front, as shown in Figure 5.12. Even if it attains to place a number of solutions along some of the disconnected areas of the optimal front, it fails in finding new solutions in the unexplored areas and in moving the existing solutions in the feasible region toward the sharp edges in the vicinity of the global optimal Pareto front.

In the next test problems the shape of the Pareto front and the distribution of the solutions are modified in such a way that the problem complexity gradually increases. The main obstacle in the previous problems was the presence of disconnected regions in the Pareto front. The problems are now made more difficult reducing the number of solutions in each segment to a single optimal solution. The third constrained problem $\mathcal{T}_{10}$ has a frontier formed by 13 disconnected point solutions, equally spaced along the locus of points previously defined. Every evolutionary strategy is able to place a non dominated solution in each regions with comparable convergence rates as

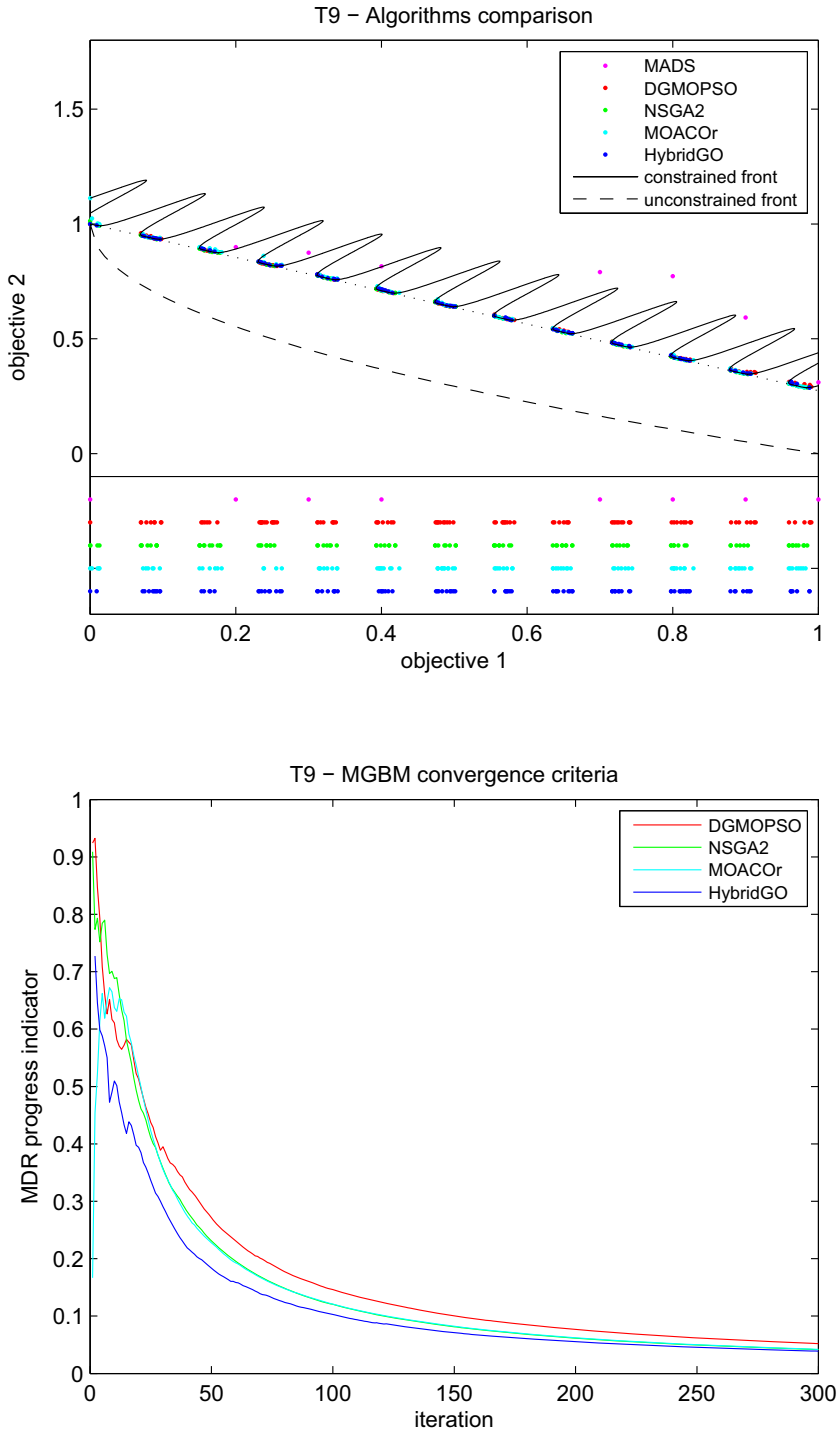


FIGURE 5.12. Pareto frontier and convergence comparison, using the MGBM criterion, of evolutionary strategies and MADS algorithms for the unconstrained multi objective optimization test problem $\mathcal{T}_9$.

shown in Figure 5.13. Only the NSGA2 algorithm fails in placing a solution in the right extreme of the Pareto front. The deterministic algorithm shows again its limit in converging to narrow disconnected optimal regions and the high computational time required by the algorithm is index of the failure of the search technique.

The forth constrained problem $\mathcal{T}_{11}$ is probably the most challenging one. The Pareto front is at the tip of a very narrow feasible region. It tests the ability of the algorithms to move between discontinuous feasible regions even far from the optimal front. As in the previous example, the evolutionary strategies, except the genetic algorithm, are able to cover homogeneously the Pareto front placing at least one solution in all the disconnected areas, with comparable values for the MDR operator, see Figure 5.14. The lack of diversification between the MDR values of the different strategies is due to the limited number of solutions in the optimal set. For the MADS algorithm there are no improvements since the problematics already identified in the previous examples are here emphasized.

In the fifth constrained problem $\mathcal{T}_{12}$ a difficulty due to the not uniform distribution of the feasible regions is introduced. MOACOr and HGO are comparable in terms of quality of the solutions returned and coverage of the front as shown in Figure 5.15, while NSGA2 and DGMOPSO missed in placing a solution in the right extreme of the Pareto front and the MADS algorithm experiences the same problematics already described.

The last two constrained test problems test the ability of the algorithms in exploring the entire search space even far away from the Pareto front. The search space is divided in disconnected feasible and unfeasible regions. During the optimization process the solutions have to overcome a number of infeasible holes to converge to the optimal Pareto front. All the algorithms are able to overcome such a hindrance and converge to the non dominated front, see Figure 5.16.
In the sixth constrained test case $\mathcal{T}_{13}$ the disconnected feasible regions are parallel to the optimal front and also the deterministic algorithm succeeds in covering the entire optimal front. This is not the case of the last test problem $\mathcal{T}_{14}$, where the disconnected feasible regions are perpendicular to the Pareto front. As it occurred before, the stochastic strategies outperformed the deterministic one, that is not able to distribute the solutions along disconnected fronts, see Figure 5.17.

5.2.1.3. *Conclusions.* The proposed test problems taken from literature cover a wide range of problematics related to convergence and diversification of the solutions from the vicinity of the optimal Pareto front to the entire search space. The selected optimization algorithms for global multi-objective mixed integer non linear programming problems have been validated and

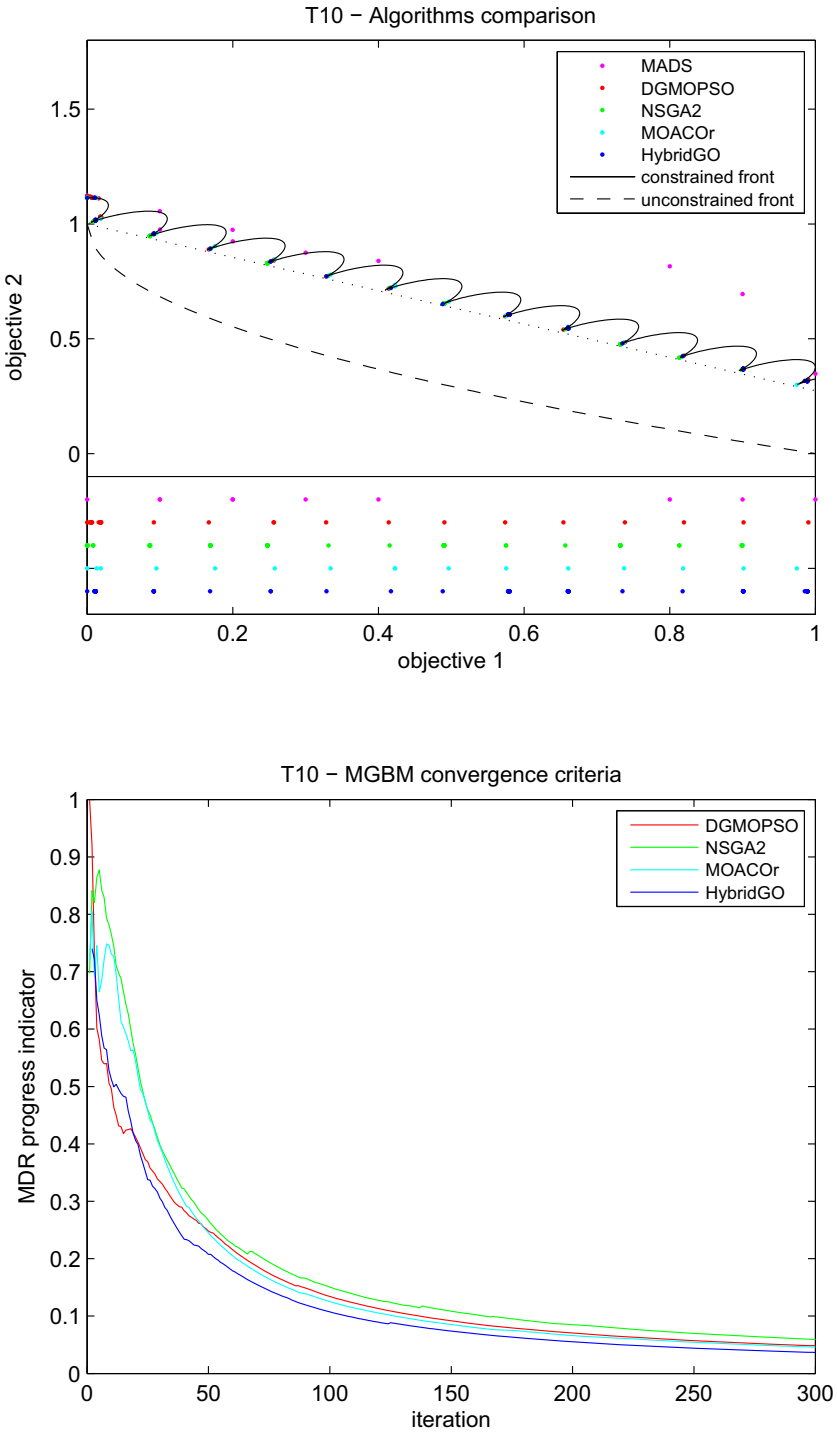


FIGURE 5.13. Pareto frontier and convergence comparison, using the MGBM criterion, of evolutionary strategies and MADS algorithms for the unconstrained multi objective optimization test problem $\mathcal{T}_{10}$.

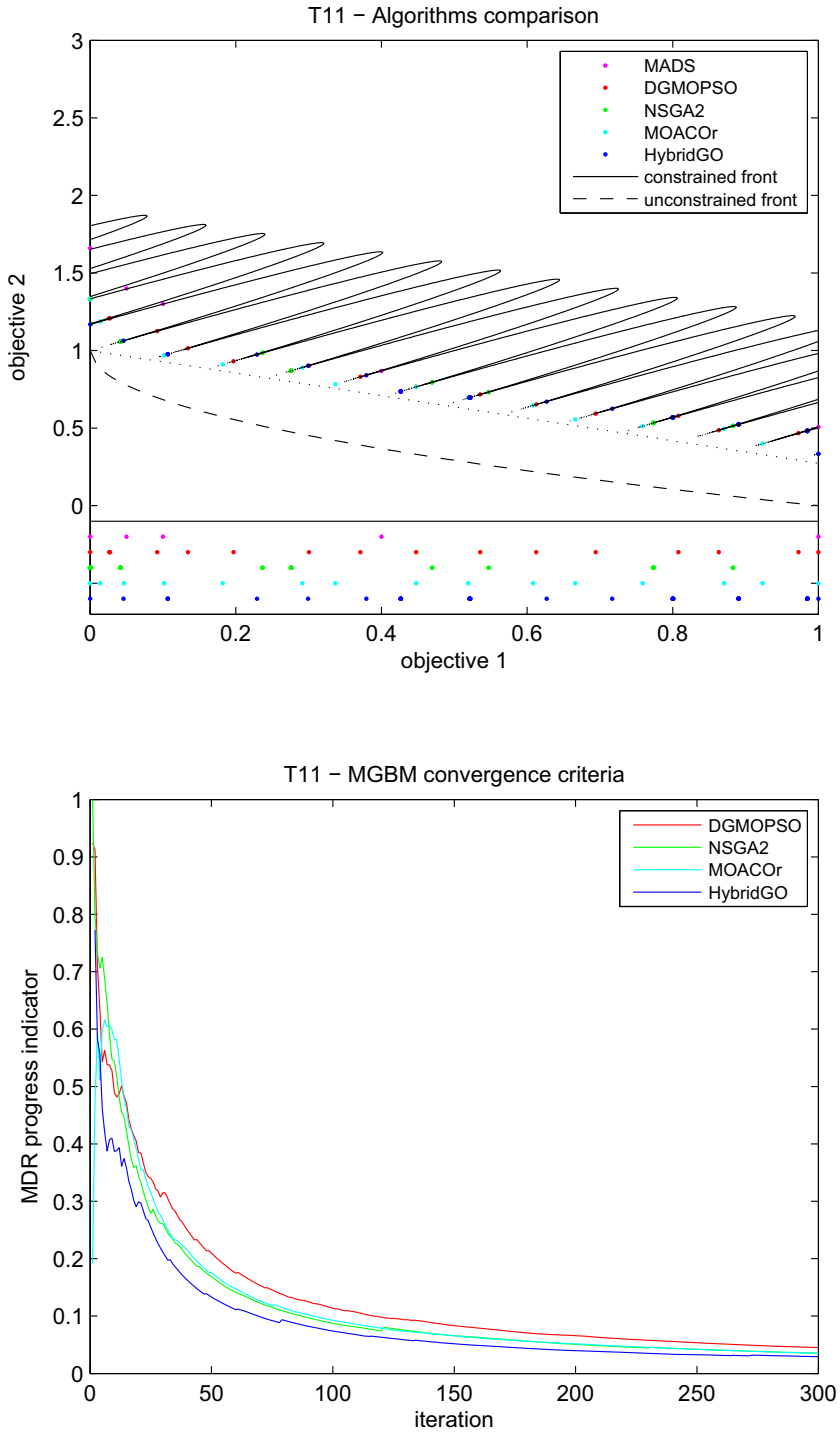


FIGURE 5.14. Pareto frontier and convergence comparison, using the MGBM criterion, of evolutionary strategies and MADS algorithms for the unconstrained multi objective optimization test problem $\mathcal{T}_{11}$.

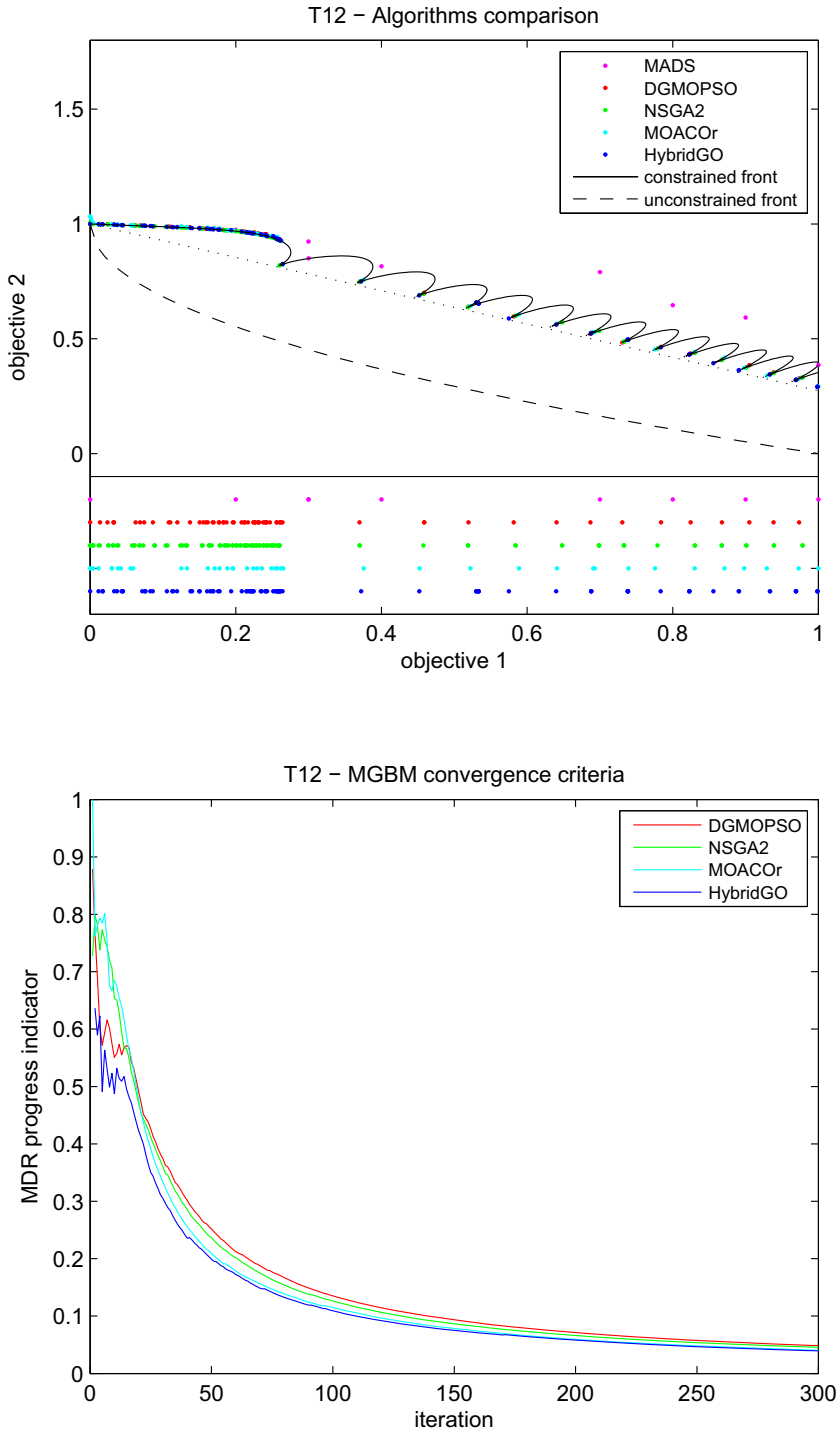


FIGURE 5.15. Pareto frontier and convergence comparison, using the MGBM criterion, of evolutionary strategies and MADS algorithms for the unconstrained multi objective optimization test problem $\mathcal{T}_{12}$.

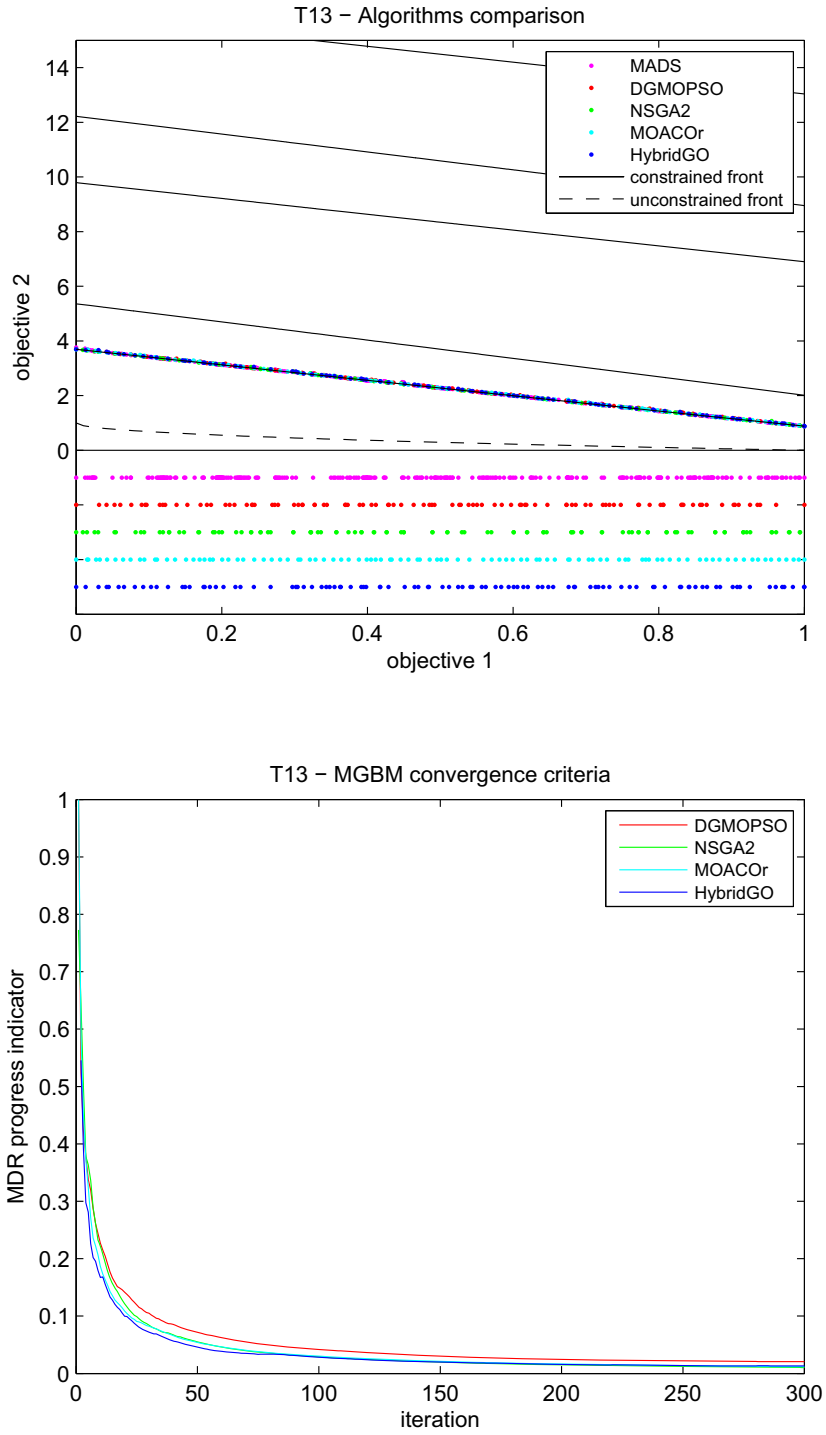


FIGURE 5.16. Pareto frontier and convergence comparison, using the MGBM criterion, of evolutionary strategies and MADS algorithms for the unconstrained multi objective optimization test problem $\mathcal{T}_{13}$.

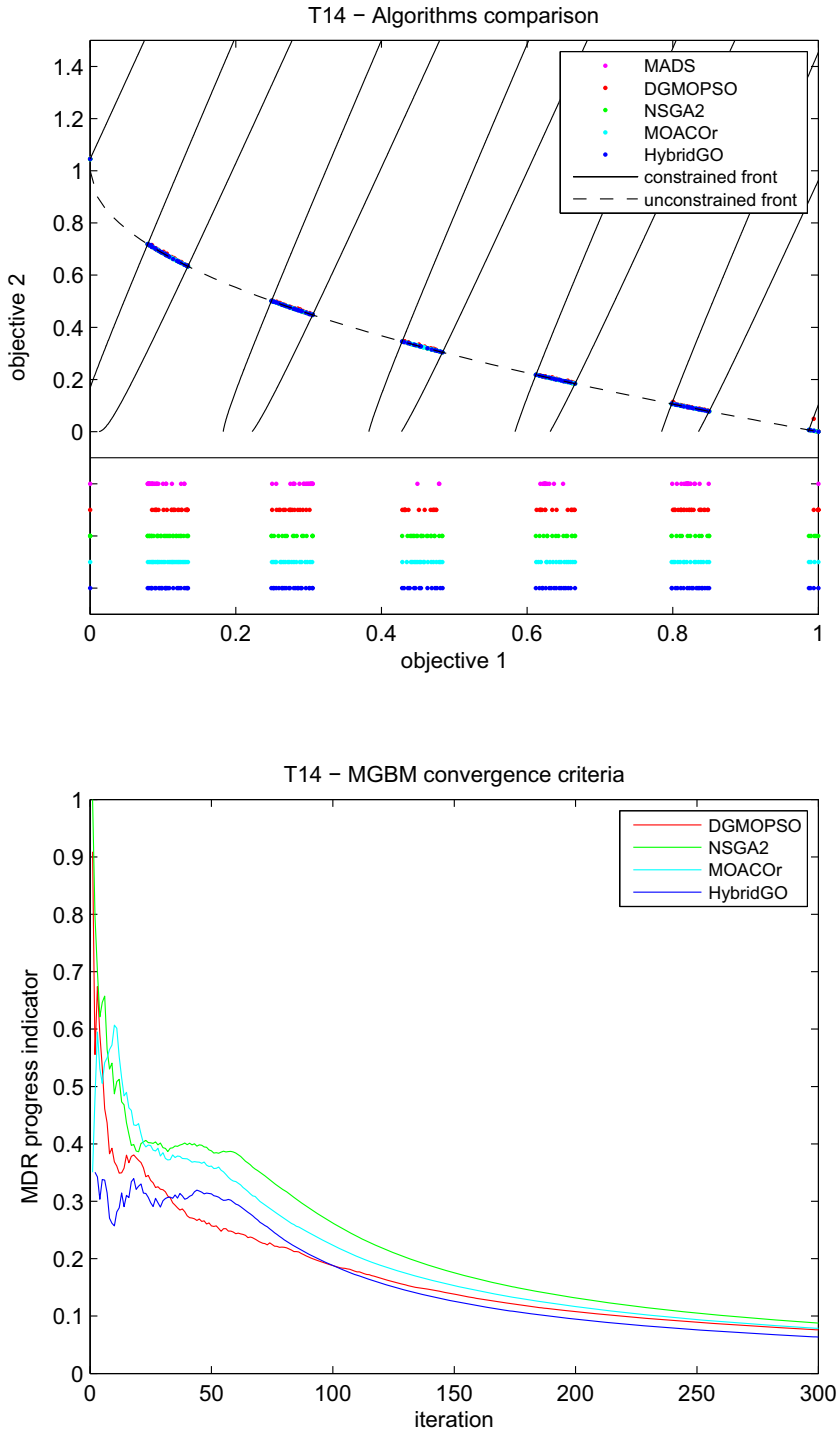


FIGURE 5.17. Pareto frontier and convergence comparison, using the MGBM criterion, of evolutionary strategies and MADS algorithms for the unconstrained multi objective optimization test problem $\mathcal{T}_{14}$.

compared. Only one problem, $\mathcal{T}_7$, is involving discrete variables, the remaining problems are continuous multi-objective NLP problems. The purpose of the benchmark tests was not to prove the ability of the optimization strategy in dealing with mixed integer non linear programming problems, but rather to compare their convergence capabilities on different challenging bi-objective optimization tests.
The algorithms were compared in terms of *performance* intended as coverage capabilities of the global optimal front, *efficiency* in terms of CPU time required for an optimization process of 30000 function evaluations and *convergence* rate evaluated through the MDR criterion.
The MADS strategy shows its limitation in solving problems with disconnected feasible search regions and disconnected optimal fronts. The algorithm is not able to maintain the diversity of the solutions between different segments of the frontier and to overcame the unfeasibility holes in the search space. On the other hand, results a very efficient strategy when compared to the other stochastic algorithms in solving multimodal optimization problems.
The HGO algorithm was designed with the purpose of exploiting the convergence capabilities of the three involved evolutionary strategies. The tests prove that the developed strategy is able to steer toward the algorithm which is behaving better on each given problem. It converges in all the test cases to the final front with a good spread of the solutions over the entire disconnected region, showing with the best rate of convergence. Even if only one strategy over the three is giving good results, the hybrid algorithm is able to favor it. None of the single stochastic strategies is solving in a comparable way the complete collection of problems with the default set of parameters. The main limitation of the HGO algorithm is its computational efficiency. This is due to the additional internal operations of recombination of the archive at every super-iteration, necessary for ordering the solutions and redistribute them back to the hybridized algorithms. This apparent computational inefficiency is irrelevant in the case the algorithm is applied for solving an optimization problem in which the computation of the model is very expensive. This is the case of MDO where the complete analysis of the vehicle design is performed in few seconds. Thus an optimization algorithm that has more computationally expensive internal routines, still on an average of seconds, are irrelevant for the total computational time where most of the computational load is taken by the evaluation of the model functions.

5.2.2. MDO Test Problems. The multi objective strategies are compared on three MDO problems for the Ariane 5 launcher test case. Different trade-offs between payload performance, total mass at launch, costs and risk are available in the model. Anyway, as outlined in [**52**], it is difficult to determine a Pareto front that represents the total vehicle mass with respect to the relative costs and risks values. This is because the employed cost model is strongly mass-dependent and hence the Pareto front often reduces

Variable	Actual	MDA	min GTOW	max Payload
$M_{\text{prop,EPC}}$	173.3	173.3	142.7(-17.6%)	144.0(-16.9%)
$M_{\text{prop,ESC-A}}$	14.4	14.4	12.6(-12.5%)	14.4(0%)
$M_{\text{prop,P241}}$	240.1	240.1	190.2(-20.8%)	271.7(+13.2%)
D_{P241}	3.05	3.05	3.6(+18.0%)	3.0(-1.6%)
GTOW [tons]	**766.4**	**770.8**	**626.5**(-18.2%)	**807.5**(+5.4%)
PL mass [kg]	**10050**	**10050**	**5025**(-50%)	**11600**(+15.4%)

TABLE 5.11. Ariane 5 ECA MDO, minimization of the GTOW mass and maximization of the payload mass results.

to single solutions. It is necessary to act on the cost model tuning case by case the specific parameters to converge to a set of non dominated solutions. For the purpose of the present work, and for a meaningful comparison of the optimization strategies selected, only performance based bi-objective optimization problems are considered.

5.2.2.1. *"Small" MDO problem.* The first optimization problem has two objectives: the minimization of the total mass at launch and the maximization of the payload mass. The set of design optimization variables is the same presented for the single objective case (Table 5.6(A)), excluding the integer variable that models the boosters structural material. The same mission to GTO is considered (see Table 5.6(B)), with optimizable payload mass. A total of 14 continuous optimization variables defines the optimization problem. The optimization strategies are executed with the default set of parameters. The total number of function evaluations is set to 100000 in the case of the stochastic algorithms, corresponding to 500 solutions in the initial set and 200 iterations, for an average computational time of about 24 hours. The maximum size of the archive is set to 100 solutions and the evolutionary algorithms are executed with three different sets of random initial solutions. The achieved Pareto fronts are compared in Figure 5.18, where the three different runs of the stochastic algorithms are marked with different symbols. The hybrid and the DGMOPSO algorithms are comparable in terms of quality of the solutions returned. The former reflects a better coverage behavior in the lower part of the front while the second one is dominating the upper side. The geometry of the two solutions, corresponding to the best achieved payload mass and GTOW, marked in Figure 5.18, is illustrated in Figure 5.19. In Table 5.11 the optimal values of the design parameters of the two solutions are compared with the actual values of Ariane 5 computed through the MDA. As expected, the most significant difference between the two solutions is related to the value of the boosters propellant mass and hence their diameter.
The NSGA2 algorithm is not able to cover the whole front. The solutions gather in the center of it without spreading over its tails, reflecting a limited exploration capability which is probably due to the presence of infeasible holes in the search space. On the contrary, MOACOr and MADS remain trapped

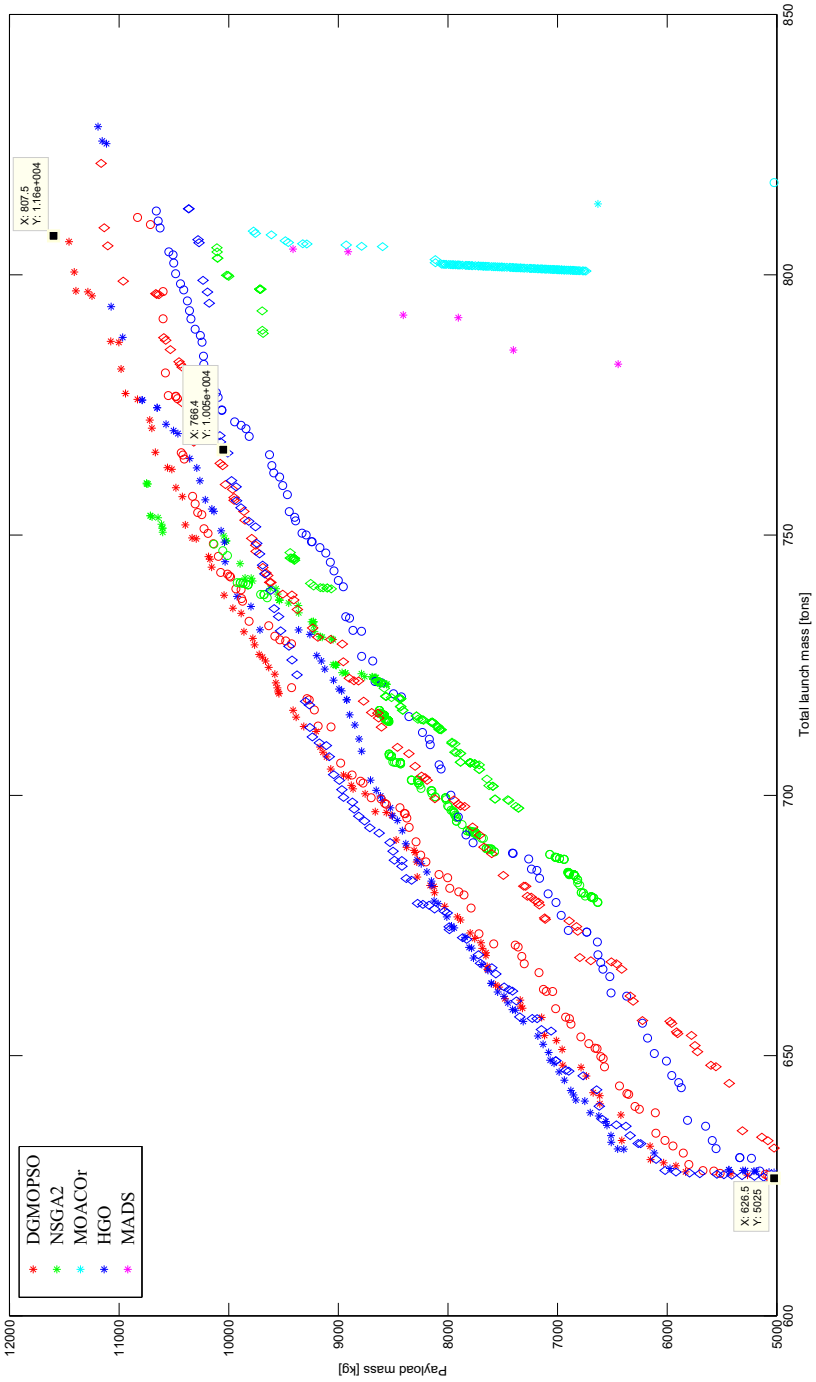


FIGURE 5.18. MDO maximization of the payload mass VS minimization of the total mass at launch for Ariane 5 ECA. Narrow bounds on propellant mass and boosters diameter.

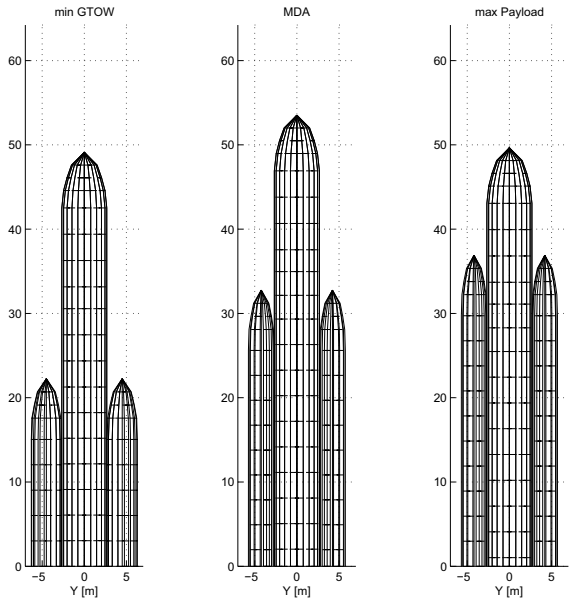


FIGURE 5.19. Pareto front extremes solutions geometry. MDO Ariane 5 minimum GTOW, maximum payload.

in local fronts and their optimal solutions are completely dominated by all the solution sets returned by the other methods. In particular MOACOr is not able to diversify the solutions along the first objective. It has been observed during the validation phase that if the MOACOr Pareto front is reducing to a point solution, the algorithm has trouble in moving away from the current non dominated solution and converging to a non dominated front. For this reason the mutation operator of NSGA2 is added to the algorithm as described in Section 3.4.2. A default percentage (10%) of the total number of design variables mutates in case of degenerated front. Increasing this value is expected to improve the algorithm behavior in diversifying the solutions along the first objective and enlarging the number of non dominated points. The MADS algorithm experiences the same problems already outlined in the analytic tests: at a comparable number of function evaluations the algorithm is able to converge to only 6 solutions, all of them dominated by the ones returned by the other algorithms.

5.2.2.2. *"Large" MDO problem.* In the small MDO problem only optimizable propellant masses are allowed, freezing all the vehicle technologies to the actual value of Ariane 5. With such a binded study it is difficult to exploit and investigate the capabilities of an MDO methodology in acting on the modifications of the vehicle design. For this purpose the next two applied examples enlarged gradually the freedom on the vehicle design, allowing to explore better, performance based, optimal solutions.
The bounds on the upper stage propellant mass are enlarged up to $\pm 50\%$ with

Variable	id	LB	UB
PLF nose length over diameter	$(L/D)_{\text{PLF}}$	0.514	1.542
LS propellant mass [tons]	$M_{\text{prop,EPC}}$	121.3	225.3
LS, EPC Nominal thrust [kN]	T_{EPC}	905.3	1681.3
LS, EPC mixture ratio	$\alpha_{\text{P,EPC}}$	5.62	6.58
LS, EPC expansion ratio	ε_{EPC}	30.0	70.0
LS, EPC chamber pressure [bars]	$p_{\text{cc,EPC}}$	89.5	120.0
LP tanks pressure [bars]	$p_{\text{tanks,EPC}}$	2.2	3.5
LP press. gas tank pressure [bars]	$p_{\text{press,EPC}}$	16.1	29.9
US propellant mass [tons]	$M_{\text{prop,ESC-A}}$	14.4	21.6
US, ESC-A Nominal thrust [kN]	$T_{\text{ESC-A}}$	31.5	94.5
US, ESC-A mixture ratio	$\alpha_{\text{P,ESC-A}}$	4.85	5.15
US, ESC-A expansion ratio	$\varepsilon_{\text{ESC-A}}$	58.2	108.0
US, ESC-A chamber pressure [bars]	$p_{\text{cc,ESC-A}}$	30.0	50.0
LP tanks pressure [bars]	$p_{\text{tanks,ESC-A}}$	2.2	3.5
LP press. gas tank pressure [bars]	$p_{\text{press,ESC-A}}$	158.2	293.8
BS propellant mass [tons]	$M_{\text{prop,P241}}$	168.1	312.2
BS diameter [m]	D_{P241}	2.44	3.66
BS, P241 Nominal thrust [kN]	T_{P241}	4476.5	8313.5
BS, P241 expansion ratio	$\varepsilon_{\text{P241}}$	7.7	14.3
BS, P241 chamber pressure [bars]	$p_{\text{cc,P241}}$	56.8	71.2

TABLE 5.12. Ariane 5 ECA MDO, large bounds, minimization of the GTOW mass. Optimization variables set.

respect to the actual value and the propulsive design of each stage and booster is left unlocked. The existing engine in the database is substituted by a new design liquid or solid rocket engine with all discrete variables frozen to the Off The Shelf (OTS) value and optimizable nominal thrust, tank pressure, chamber pressure, expansion ratio and mixture ratio. The geometry of the vehicle is left varying allowing a range of $\pm 50\%$ on the Payload Fairing (PLF) length over diameter ratio. A total of 20 continuous variables related to the vehicle design and 10 variables for controlling the trajectory (the same already defined in Table 5.6(B)) defines the final optimization problem. See Table 5.12 for a detailed list of the optimization variable set. The bi-objective problem is to minimize the mass at launch and maximize the payload mass, and it is solved with the stochastic evolutionary strategies and the deterministic derivative free algorithm MADS. The stochastic algorithms are performed in three different runs, with the default set of parameters, on an initial solution set of 500 elements for a maximum number of 200 iterations. A total of 100000 function evaluations are needed to complete the optimization process, that

Variable	MDA	min GTOW	max Payload
$(L/D)_{\text{PLF}}$	1.03	0.96 (-6.0%)	0.84 (-18.4%)
$M_{\text{prop,EPC}}$	173.3	121.3 (-30%)	190.7 (+10%)
T_{EPC}	1293.3	1152.8 (-10.9%)	1390.9 (+7.5%)
$\alpha_{\text{P,EPC}}$	6.1	6.35 (+4.0%)	5.9 (-3.2%)
ε_{EPC}	59.5	47.3 (-20.5%)	47.6 (-20.0%)
$p_{\text{cc,EPC}}$	115.0	116.5 (+1.3%)	113.4 (-1.4%)
$p_{\text{tanks,EPC}}$	3.0	3.3 (+10%)	2.2 (-26.7%)
$p_{\text{press,EPC}}$	23.0	23.1 (+0.4%)	24.1 (+4.8%)
$M_{\text{prop,ESC-A}}$	14.4	16.0 (+11.1%)	20.8 (+44.4%)
$T_{\text{ESC-A}}$	63.0	68.7 (+9.0%)	94.5 (+50%)
$\alpha_{\text{P,ESC-A}}$	5.0	5.01 (+0.2%)	5.05 (+1.0%)
$\varepsilon_{\text{ESC-A}}$	83.1	65.6 (-21.1%)	73.7 (-11.3%)
$p_{\text{cc,ESC-A}}$	37.0	45.1 (+21.9%)	46.3 (+25.1%)
$p_{\text{tanks,ESC-A}}$	3.0	2.5 (-16.7%)	2.7 (-10.0%)
$p_{\text{press,ESC-A}}$	226.0	158.2 (-30.0%)	239.0 (+5.8%)
$M_{\text{prop,P241}}$	240.1	168.1 (-30.0%)	285.5 (+18.9%)
D_{P241}	3.05	2.53 (-17.0%)	3.5 (+14.7%)
T_{P241}	6395.0	4476.5 (-30.0%)	7607.0 (+18.9%)
$\varepsilon_{\text{P241}}$	11.0	9.3 (-15.4%)	11.5 (+4.5%)
$p_{\text{cc,P241}}$	64.0	56.8 (-11.2%)	59.4 (-7.2%)
GTOW [tons]	**770.8**	**545.3** (-28.8%)	**900.9** (+17.5%)
Payload [kg]	**10050**	**5446.5** (-45.8%)	**13382** (+33%)

TABLE 5.13. Ariane 5 ECA MDO, large bounds, minimization of the GTOW mass. Optimal values.

corresponds to nearly 24 hours of computation time. The same number of function evaluations is set for the black box deterministic strategy as stopping condition. The Pareto fronts achieved by the different methods are compared in Figure 5.20 with the actual value of Ariane 5 performances represented by the black dot.

The second run of the hybrid algorithm is dominating all the solutions returned by the other algorithms and is able to cover widely the optimal front. The design optimal values of the two extreme solutions are reported in Table 5.13 and their geometries are shown in Figure 5.21.

If the solutions are compared with the ones obtained in the "small" MDO test case, it is observed that the optimization algorithm has more freedom in redesigning the performance of the engines and it converges to a front that

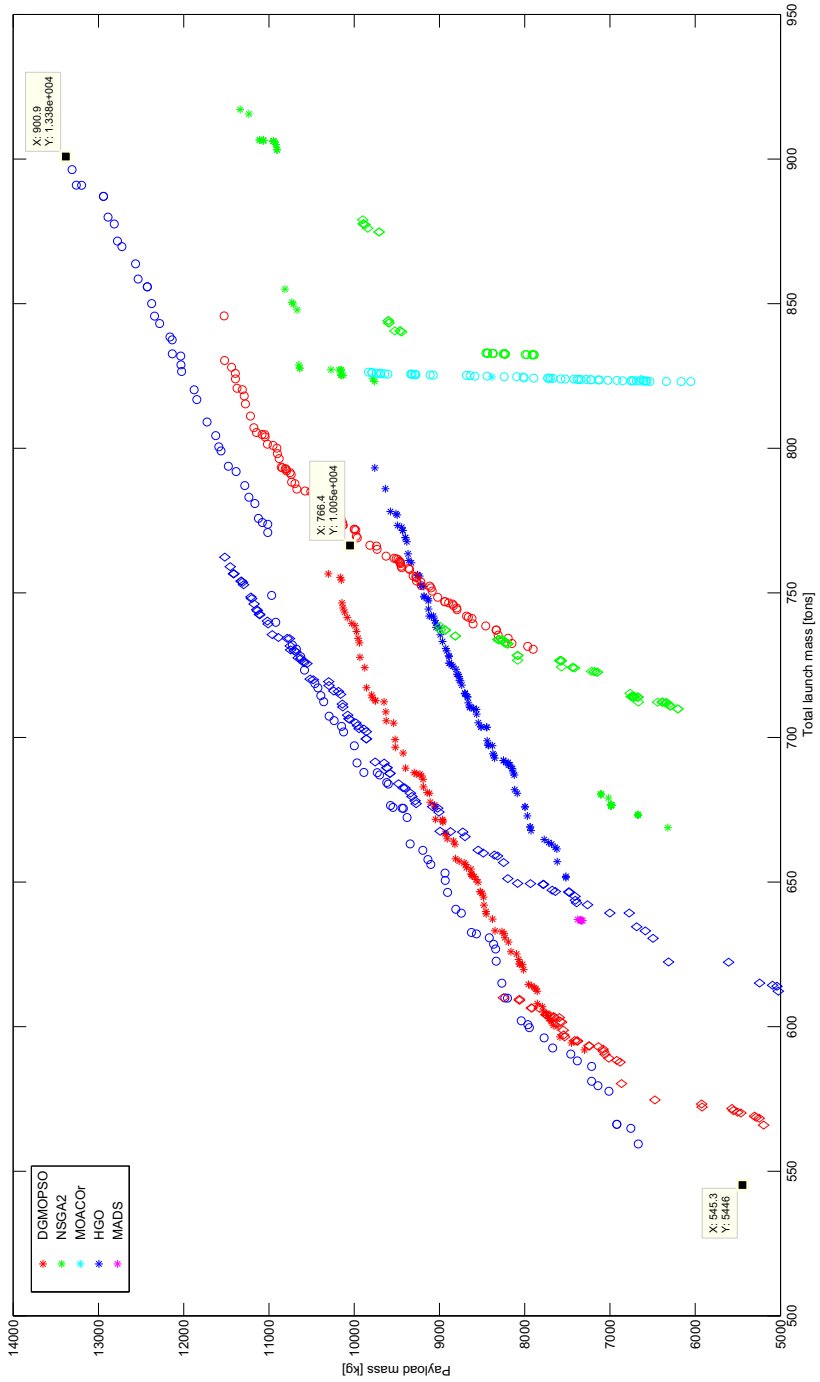


FIGURE 5.20. MDO large bounds, minimization of the GTOW and maximum of the payload mass. Pareto frontiers comparison

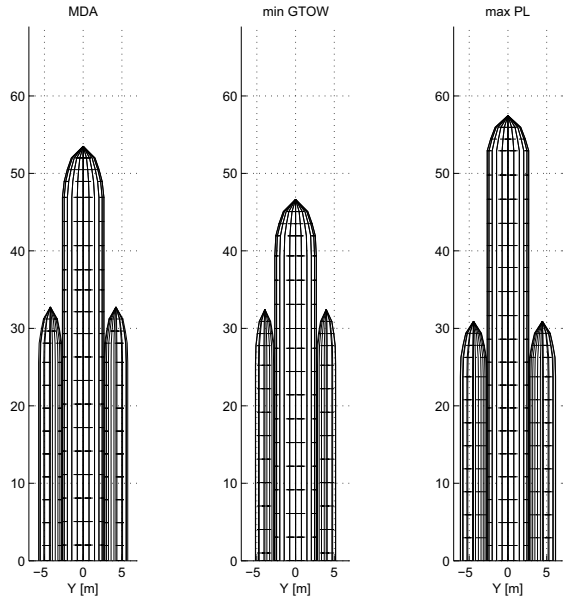


FIGURE 5.21. Pareto frontier extremes solutions geometry. MDO large bounds minimum GTOW, maximum payload.

		min GTOW	max Payload
"Small" MDO	**GTOW [tons]**	**626.5** (-18.2%)	**807.5** (+5.4%)
	PL mass [kg]	**5025** (-50%)	**11600** (+15.4%)
"Large" MDO	**GTOW [tons]**	**545.3** (-28.8%)	**900.9** (+17.5%)
	Payload [kg]	**5446.5** (-45.8%)	**13382** (+33%)

TABLE 5.14. Ariane 5 ECA "small" and "large" MDO problems, optimal values comparison.

completely dominates the previous one. In Table 5.14 are reported for comparison the extreme points objective values of the two fronts. In particular, from Table 5.13 it is observed that, for minimizing the total mass at launch, the optimizer acts prominently on the modification of the boosters. The mass of propellant of each booster is reduced by 30% and the engine redesigned with lower nominal thrust and smaller diameter, constituting the greatest saving of the total mass at launch together with the reduction of 52 tons of the lower stage propellant mass. The upper stage has minor changes with respect to the previous two components, mainly affecting the chamber and pressurization gas tank pressure.
For maximizing the payload mass, the upper stage and boosters are the components with the most noteworthy modifications. The respective propellant masses and nominal thrusts enlarged up to respectively 19% and almost 50%.

Greater changes are detected also in the geometry of the payload faring that reduces its length, and on the diameter of the boosters that adapts to the increase of the propellant mass.
As observed in the previous test case both DGMOPSO and HGO achieve well spread optimal solutions, although the latter one is able to converge to a complete non dominated front. The genetic and the ant colony algorithms fail in returning competitive solution sets in any of the optimization runs. The ant colony strategy is experiencing the same problems in differentiating the solutions along the first objective while the genetic algorithm is converging to smaller and dominated areas of the search space limiting its exploration capabilities on confined "islands". The MADS algorithm is finding a reduced set of 8 solutions that gather in a small dominated region of the feasible space.

5.2.2.3. *Free Architecture MDO problem.* To prove the ability of the MDO approach in exploring different launcher design configurations and to include discrete variables also for the multi-objective tests, the same GTO mission of Ariane 5 used in the previous examples is freed from any constraints on the launcher architecture. A total of 45 optimization variables (4 for the system design, 16 for the lower stage, 8 for the upper stage, 7 for the boosters and 10 for the trajectory) define the new optimization problem. In Table 5.15 only the most representative variables are reported, the specific variables related to the design of a new liquid rocket engine for the lower stage and the upper stage propulsive design parameters are omitted.

Variable	Id	Bounds
Boosters configuration	*Config*	{No boosters, Circumferential}
Number of Boosters	N_{boosters}	{2, 3, 4}
Length over diameter PLF	$(L/D)_{\text{PLF}}$	[0.514, 1.542]
Pad interface type	PIC	{boosters, core}
LS diameter equal to US	$D_{\text{LS=US}}$	{true, false}
LS diameter [m]	D_{LS}	[4.0, 6.5]
LS number of engines	$N_{\text{engines,LS}}$	{1,2,3,4}
LS nominal thrust [MN]	T_{LS}	[0.3, 3.0]
LS propellant mass [tons]	$M_{\text{prop,LS}}$	[100.0, 250.0]
LS Off the Shelf	OTS_{LS}	{true, false}
LS Off the Shelf Id	$OTS_{\text{LS,Id}}$	{RS-68,Vulcain-2}
LS tanks arrangements	$TanksArr_{\text{LS}}$	{Ox,Fuel after}
LS tanks type	$Tanks_{\text{LS}}$	{common bulkhead, separate bulkhead}
LS main structural material	SM_{LS}	{Al7075, Al-Li}
LS secondary structural material	sm_{LS}	{Al7075, CFRP}

US, diameter equal to PLF	$D_{\text{US=PLF}}$	{true, false}
US diameter [m]	D_{US}	[4.0, 5.4]
US number of engines	$N_{\text{engines,US}}$	{1,2}
US propellant mass [tons]	$M_{\text{prop,US}}$	[14.0, 40.0]
US Off the Shelf Id	$OTS_{\text{US,Id}}$	{RL10B-2, RD-0146, Vinci, HM-7B, Merlin1V}
US tanks type	$Tanks_{\text{US}}$	{common bulkhead, separate bulkhead, enclosed}
US main structural material	SM_{US}	{Al7075, Al-Li}
US secondary structural material	sm_{US}	{Al7075, CFRP}
BS diameter	D_{BS}	[1.5, 3.5]
BS nominal thrust [MN]	T_{BS}	[2.0, 9.0]
BS propellant mass [tons]	$M_{\text{prop,BS}}$	[80.0, 300.0]
BS chamber pressure norm	$p_{\text{cc,BS}}$	[0.2, 0.8]
BS nozzle expansion ratio	ε_{BS}	[7.0, 20.0]
BS main structural material	SM_{BS}	{Steel,CFRP}
BS secondary structural material	sm_{BS}	{Al7075,CFRP}

TABLE 5.15. MDO free architectural design, minimization of the GTOW mass and maximization of the payload mass optimization variables.

Only the stochastic strategies are compared since the MADS algorithm is not able to solve the optimization problem, probably due to the high number of optimization variables involved, that is a limit of the strategy as already outlined in Section 3.4. The evolutionary algorithms have been executed with the default set of parameters for a total number of 200000 function evaluations, that correspond to 500 solutions in the initial set, 400 iterations and an average computational time of 58 hours. The maximum size of the archive is set to 100 solutions and the algorithms are executed for three different sets of random initial solutions.
The achieved Pareto fronts are reported in Figure 5.22. The third run of the hybrid algorithm corresponds to the best set of non dominated solutions. The two solutions matching the two extremes of the front are reported in Figure 5.23 and the corresponding optimal design parameter values are listed in Table 5.16.
The optimal front is disconnected. The solutions belonging to the two disconnected parts differ for the number of boosters. The maximum payload performance is obtained with a circumferential configuration of 3 boosters

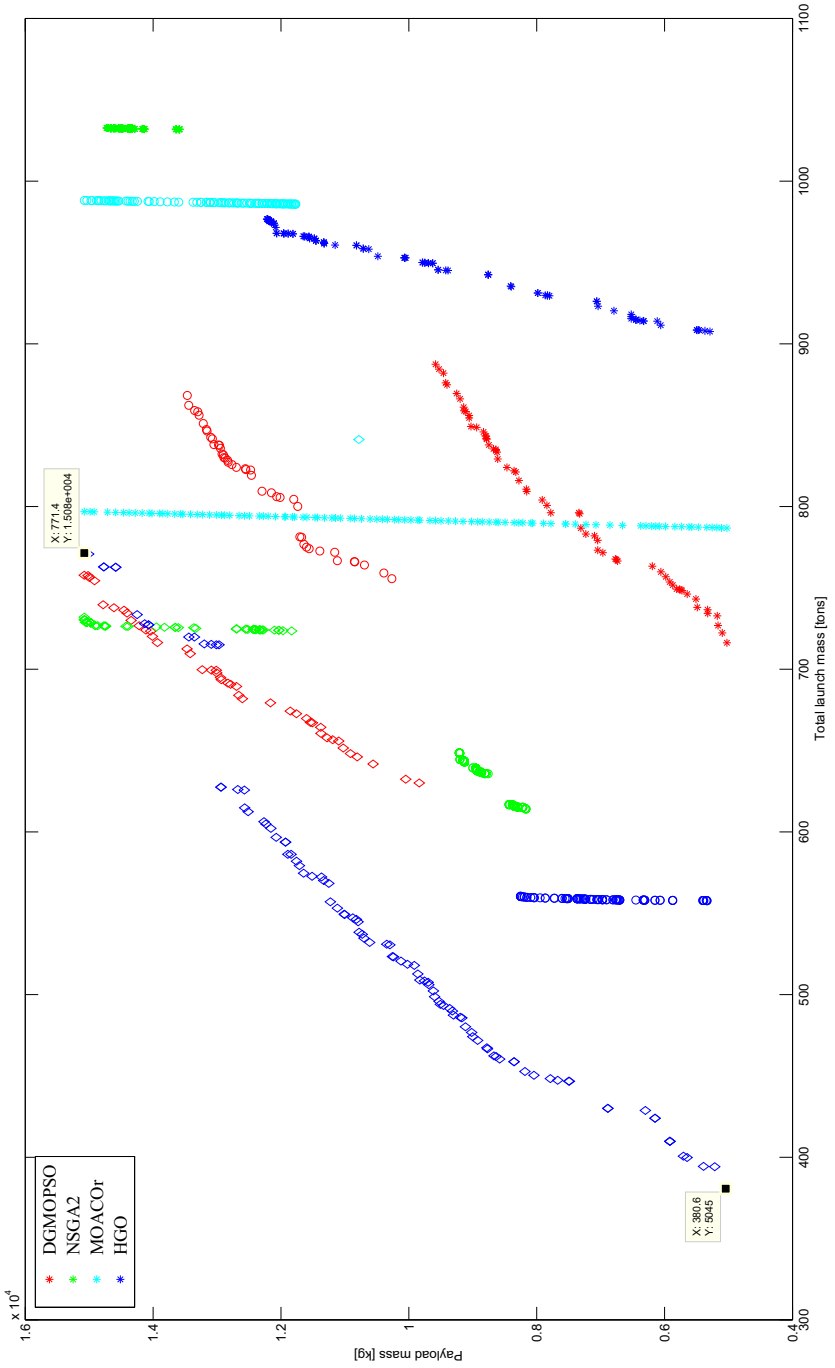


FIGURE 5.22. MDO Free Architecture, minimization of the GTOW and maximum of the payload mass. Pareto frontiers comparison

Variable	min GTOW	max PL
Config	Circumferential	Circumferential
N_{boosters}	2	3
$(L/D)_{\text{PLF}}$	0.63	0.514
PIC	boosters	core
$D_{\text{LS=US}}$	true	true
D_{LS}	5.4	5.4
$N_{\text{engines,LS}}$	2	2
T_{LS}	2.6	2.6
$M_{\text{prop,LS}}$	144.1	204.1
OTS_{LS}	true	true
$OTS_{\text{LS,Id}}$	Vulcain-2	Vulcain-2
$TanksArr_{\text{LS}}$	Fuel after	Fuel after
$Tanks_{\text{LS}}$	Separate bulkhead	Separate bulkhead
SM_{LS}	Al7075	Al-Li
sm_{LS}	CFRP	CFRP
$D_{\text{US=PLF}}$	true	true
D_{US}	5.4	5.4
$N_{\text{engines,US}}$	2	2
$M_{\text{prop,US}}$	26.7	27.5
$OTS_{\text{US,Id}}$	RD-0146	RD-0146
$Tanks_{\text{US}}$	Separate bulkhead	Separate bulkhead
SM_{US}	Al-Li	Al-Li
sm_{US}	CFRP	CFRP
D_{BS}	2.9	2.9
T_{BS}	2.8	4.2
$M_{\text{prop,BS}}$	80.0	147.9
$p_{\text{cc,BS}}$	0.3	0.58
ε_{BS}	15.5	18.5
SM_{BS}	CFRP	CFRP
sm_{BS}	CFRP	CFRP
GTOW [tons]	**380.6**	**771.4**
Payload [kg]	**5044.7**	**15075**

TABLE 5.16. MDO free architectural design, minimization of the GTOW mass and maximization of the payload mass optimization variables.

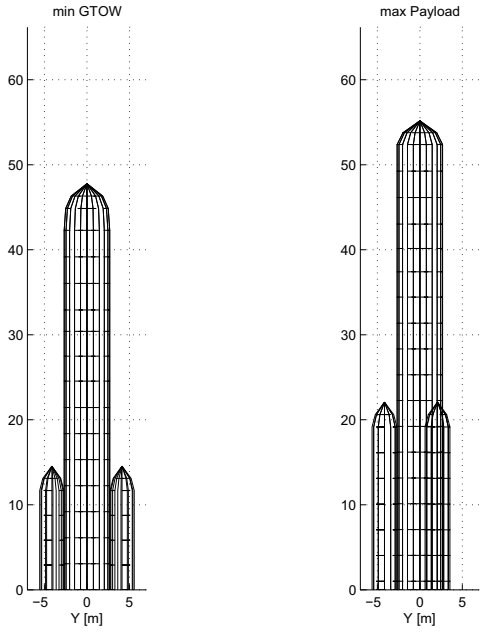


FIGURE 5.23. Pareto front extremes solutions geometry. MDO Free architecture minimum GTOW, maximum payload.

with 148 tons of propellant each, while the minimum mass at launch configuration is obtained with 2 boosters provided by 80 tons of propellant mass each. The geometry of the two different booster sets adapts in length to the amount of propellant used, while the size of the diameter is the same for both solutions. Upper stage and lower stage have the same propulsive system composed of two engine each, respectively RD-0146 and Vulcain-2, 60 tons more of propellant are necessary in the lower stage for reaching the maximum payload value while the diameter is equal in both solutions. The payload faring is slightly longer for the minimum total mass solution and the difference in the employed materials is observed only in the main structural material of the lower stage that employs Al-Li alloy instead of aluminum.

5.2.2.4. *Conclusions.* The multi-objective global optimization techniques for MINLP have been tested on applicative MDO problem acting on redesigning the Ariane 5 vehicle with a gradual enlargement of the design freedom. The obtained results, showing the ability of MDO to bring sound modifications on the existing test launcher, are a further validation of the effectiveness of the MDO technique in taking advantage of the concurrent optimization of the vehicle subsystems to reach good optimal solutions.

As already experienced in the validation of the algorithms on the analytic test cases, the stochastic strategies are the most promising techniques for solving the challenging MDO problem. The presence of integer and categorical design variables lead to a disconnected optimal front, hardly handled by the

deterministic technique. Moreover a total of 48 constraints, mostly regarding the vehicle design, defines the optimization problem with a resulting discontinuous feasible space. Among all the stochastic algorithms DGMOPSO and HGO achieved the best results for the proposed problems. The latter, in particular, was able to return in two of the test cases a Pareto front that dominates the solution sets returned by all the other algorithms in all the different runs. This is a further validation of the hybrid technique of exploiting the convergence capabilities, of each of the algorithms involved in the hybridization, through their collaboration along all the optimization process. The major drawback of using stochastic techniques is their computational efficiency. The time required for solving the proposed MDO problems can be drastically reduced exploiting the OpenMP parallelization of every evolutionary strategy already implemented in the tool. The computational time will be cut down proportionally to the number of processors available in the used machine (nearly linear speed-up).

5.3. Validation of the Trajectory Optimization Subproblem

The ascent trajectory optimization problem has been studied separated from the MDO problem since one of the capability of the software is to perform stand alone trajectory optimization. For details about the software operational modes the reader may refer to Appendix B. Moreover, it has already been mentioned in Section 2.4, that the trajectory optimization subproblem can be inserted in a nested optimization loop within the multidisciplinary analysis. To this end the validation of the optimization process is performed for the actual Ariane 5 design and for the same design computed through the multidisciplinary analysis.

The trajectory optimization problem is an optimal control problem that through direct methods can be transformed into an NLP problem. Two single objective optimization approaches are compared, the deterministic NLP solver WORHP based on local SQP techniques and the global stochastic PSO algorithm. Besides that, in light of the application of a NOL MDO architecture, also two different methods for the integration of the trajectory have been tested and compared with the objective of attaining fast and reliable payload assessments. The integration of the equation of motion with classical Runge-Kutta methods is compared with a semianalytic method that aims of computing analytic solutions of the differential equations system on smaller intervals.

Three main figures of merit of the trajectory optimization process have been considered:

Robustness in terms of repeatability of the results. This can be evaluated varying the initial guess, the algorithm parameters (in case of deterministic strategies) or the seed (in case of stochastic strategies) and verifying that the same or comparable optimal solutions are obtained,

Performance in terms of optimal value and constraints fulfillment,

Computational efficiency considering the average time required to reach the optimal solution.

Although all these aspects are extremely important for a successful trajectory optimization tool, robustness seems the most critical within MDO problems. In fact, a trajectory framework that does not allow for reliable evaluation of the payload performance may artificially bias the multidisciplinary optimization toward design solutions, for which the trajectory optimization proceeds better.

5.3.1. Modeling. For a detailed description of the trajectory model please refer to Section 4.2. In this section just the model enhancements and modifications in the optimization problem definition made during the testing phase are presented, according to the optimization strategy in use. Two different optimization approaches, a local deterministic and a global stochastic, are compared. Although the smoothness of the model is not an issue when using global optimization algorithms, particular attention has to be paid to this aspect when applying gradient-based methods such as WORHP. Smoothness is intended in this case as the absence of any discontinuity in objective and constraints as well as the introduction of the possible minimum number of local minima and disconnected feasible regions. For the trajectory models the objective function is a simple linear function, because the payload mass is maximized through a scaling factor that is one of the design variables of the optimization problem, therefore it is smooth by nature. The constraint functions are instead highly not linear. Two different types of constraints are applied: the matching of the target orbital parameters at the end of the ascent flight (semiaxis a, eccentricity e and inclination i), and path constraints valid at each time instant representative of the maximum loads (heat flux q_{heat}, axial acceleration n_{ax}, dynamic pressure q_{dyn}, angle of attack α_{AoA}, static controllability). The final orbit constraints were found to be extremely not smooth with the model developed for the optimization with the PSO algorithm. This issue was highlighted from the very poor robustness properties initially shown by the trajectory model when applying a local method. In particular, very small variations in any of the launcher design parameters or in the initial guess would result in a completely different optimal solution, indicating the presence of a large number of local minima with very small regions of attraction. Path constraints on the other hand didn't appear as an issue for the optimization problem, probably due to their dependency on a smaller set of optimization variables. The problem was identified by plotting the constraints surfaces as a function of two of the most influential optimization variables: the initial pitch-over angle and the payload scaling factor (which is also the optimization's objective function) fixing the remaining optimization variables to the optimal values returned by WORHP and running different trajectory simulations on the grid nodes. The main issue highlighted by the analysis is shown in Figure 5.24 for the flight of Ariane 5 ECA toward a standard GTO. The final orbit semiaxis presents a flat region

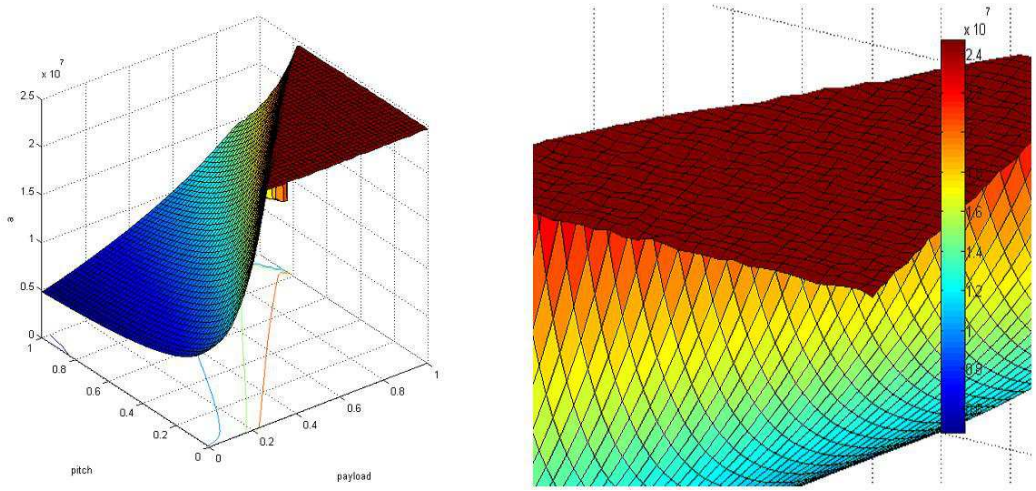


FIGURE 5.24. Final orbit's semiaxis constraint surface as a function of initial pitch-over angle and payload mass for Ariane 5 ECA to GTO. The plot is obtained by freezing all other optimization variables.

where the target value is matched. Since the integration of the equations of motion is stopped as soon as the required orbital energy is reached, this region is large but extremely "bumpy". In fact, there are hundreds of local minima and maxima of the constraint surface in the feasible area, due to the discrete nature of the variable step size integration process. This results in different instants of integration stopping, sometimes within and sometimes out of the allowed tolerance on the semiaxis. Such a model leads to a disconnected feasible region.
For different initial guesses or when varying the algorithm parameters WORHP gets stuck in different local minima as shown in Figure 5.25 where the contour lines of the semiaxis function and the corresponding disconnected feasible area are visualized. The set of feasible solutions is therefore discontinuous, so that each time the optimization is started with different settings, the final solution can end up being anywhere in the feasibility region without a guarantee on reaching the best feasible point (as it happens in Figure 5.25 where the optimal solution returned by WORHP is represented by the red cross in the feasible space). While the presented model is suitable and more efficient for a global stochastic optimization approach, since the feasible area is larger, the model has to be redefined for local gradient based algorithms. It can be corrected by simply avoiding the stopping of the integration at the target orbital energy, imposing that the propellant of the upper stage is always fully depleted. It implies a narrowing of the feasible region, which is inefficient for global strategies, but allows obtaining a smoother model and reduces the multimodality of the optimization problem functions. As shown in Figure 5.26, WORHP is capable of pushing the solution as far to the right as possible, so that the achieved optimum (marked with the red cross) corresponds to the best payload solution in the feasible set. To recover the wider feasible region

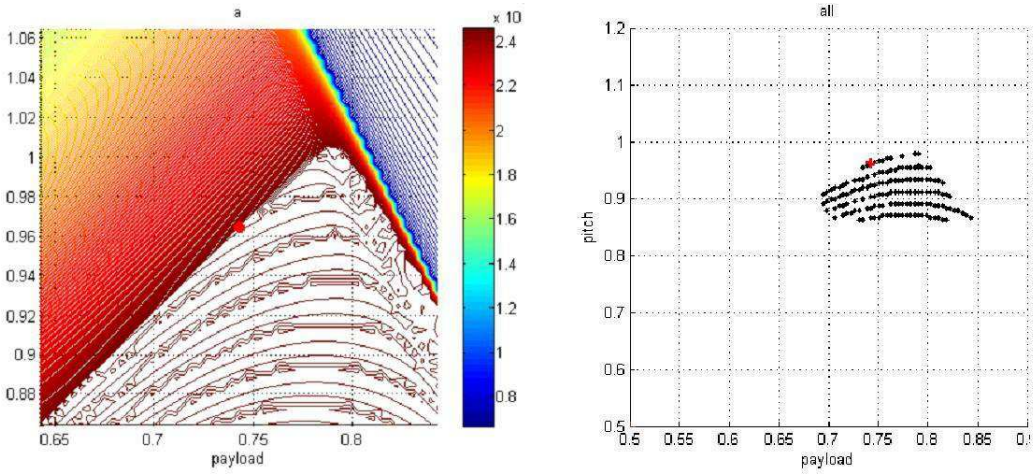


FIGURE 5.25. Left: contour plot of the constraint surface of Figure 5.24. Right: set of feasible solutions obtained varying initial pitch-over and payload. The red solution at $x = y = 0.5$ is the initial guess, the red solution in the feasibility region is the solution obtained by WORHP, clearly a local optimum.

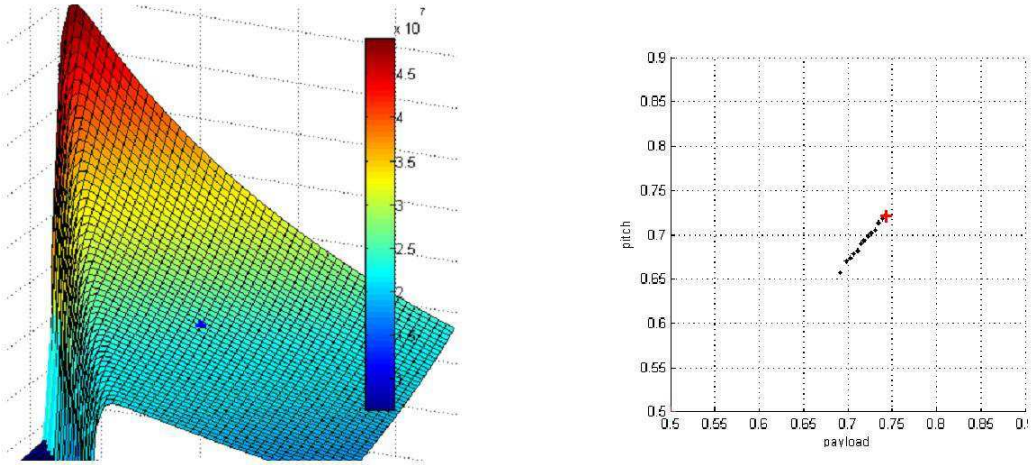


FIGURE 5.26. Left: 3D plot of the semiaxis constraint surface for Ariane 5's flight to GTO after the removal of the integration stopping condition ("smooth" model). Right: set of feasible points, initial guess at $x = y = 0.5$ and globally optimal solution returned by WORHP for the "smooth" model.

found with the stopping criteria and maintain the regularity of the problem an additional optimization variable can be added to the problem. It models the propellant left in the last stage allowing variable time of flight. Naturally the optimum still resides in the solution with no fuel left over but the feasible search space will be enlarged to the original one.

The distinction between equality and inequality constraints has been another major issue registered in the original trajectory model designed for the PSO algorithm where all constraints are treated as inequality constraints allowing given tolerances on the final orbital parameters. This constraints relaxation is necessary in case of a global optimization approach, too coarse to allow the precise matching of equality constraints. However, WORHP is capable of handling both equality and inequality constraints, and the choice among the two types does influence the behavior of the optimization algorithm. Treating an equality constraint as inequality enlarges the corridor of acceptable solutions. This can be done by adding, to the tolerance set by the user, the tolerance employed by the algorithm for feasibility determination. In particular, inequality (g_i, $i = 1, 2, 3$) and equality (h_i, $i = 1, 2, 3$) constraints on the final target orbit are defined as follow

$$\begin{aligned} g_1(X) &= \frac{a(X) - a_{\text{target}}}{a_{\text{tol}}}; \quad h_1(X) = \frac{a(X) - a_{\text{target}}}{a_{\text{norm}}} \\ g_2(X) &= \frac{e(X) - e_{\text{target}}}{e_{\text{tol}}}; \quad h_2(X) = \frac{e(X) - e_{\text{target}}}{e_{\text{norm}}} \\ g_3(X) &= \frac{i(X) - i_{\text{target}}}{i_{\text{tol}}}; \quad h_3(X) = \frac{i(X) - i_{\text{target}}}{i_{\text{norm}}}, \end{aligned}$$

where X is the vector of optimization variables; $a(X)$, $e(X)$, $i(X)$ are the orbital parameter functions, a_{target}, e_{target}, i_{target} are the target orbit semiaxis, eccentricity and inclination values, a_{tol}, e_{tol}, i_{tol} are the tolerance allowed by the user and a_{norm}, e_{norm}, i_{norm} are the scaling factors. The constraints bounds are therefore defined as

$$g_i(X) \leq 0; \quad h_i(X) = 0; \quad i = 1, 2, 3.$$

On top of these definitions, WORHP further defines a tolerance on the constraints values, corresponding to the parameter $\varepsilon_{\text{feas}}$. In order to ensure a fair comparison between the statements of inequality and equality constraints, the following relations must therefore hold

$$a_{\text{norm}} = \frac{a_{\text{tol}}}{\varepsilon_{\text{feas}}}; \quad e_{\text{norm}} = \frac{e_{\text{tol}}}{\varepsilon_{\text{feas}}}; \quad i_{\text{norm}} = \frac{i_{\text{tol}}}{\varepsilon_{\text{feas}}}$$

so that, when WORHP accepts the constraints violation on a given parameter in the equality case, the error is lower or equal to the allowed tolerance. The computation of the normalization factor is done automatically internally to the call to the local optimization routine, according to the above equations. Only the tolerance on the accuracy and the tolerance on the feasibility need to be defined by the user.

In the results presented below, the modeling and problem definition modifications presented before apply according to the optimizer in use. Summarizing, for the stochastic strategy PSO the stopping criteria on reaching the target semiaxis is enabled and the equality constraints are treated as inequality constraints allowing a predefined tolerated error on the target orbital parameters. On the contrary, for the WORHP optimizer the stopping condition is

Variable	id	LB	UB
Payload scaling factor [-]	x_{PL}	0.5	1.5
Take off pitch deviation [deg]	$\Delta\theta_{\mathrm{PO}}$	1	5
Take off pitch over duration [s]	Δt_{PO}	2	10
Take off pitch over decay time [s]	$\Delta t_{\mathrm{PO,decay}}$	1	5
Take off pitch over heading angle [deg]	ψ_{PO}	-10	10
Atmospheric, pitch deviation [deg]	$\Delta\theta_{\mathrm{EPC}}$	-10	20
Atmospheric, yaw deviation [deg]	$\Delta\psi_{\mathrm{EPC}}$	-10	10
Exoatmospheric, yaw deviation [deg]	$\Delta\psi_{\mathrm{ECA}}$	-10	10
BTL initial pitch [deg]	$\Delta\theta_{\mathrm{BTL,i}}$	-50	50
BTL final pitch [deg]	$\Delta\theta_{\mathrm{BTL,f}}$	0	50
BTL parameter [-]	ξ_{BTL}	-1	1

TABLE 5.17. Ariane 5 ECA MDO, Trajectory optimization variables set for GTO mission.

disabled and the integration of the trajectory stops after the complete depletion of the upper stage propellant. Moreover, for the WORHP case, the equality constraints are treated as such and scaled by a normalization factor computed in compliance with the allowed error on the final orbit and the feasibility tolerance of the optimizer.
Furthermore, as already discussed in the MDO problem, all optimization variables are scaled between 0 and 1 and the path constraints according to a proper reference value.

5.3.2. Test Case. The ascent trajectory has been optimized with both the global PSO and the local WORHP algorithms for the test case of Ariane 5 ECA launched from Kourou to GTO. The trajectory model is divided in 5 phases (see Section 4.2.2 for a detailed explanation): vertical take off, pitch over maneuver, first stage flight with boosters, first stage flight without boosters and second stage flight. The throttle of the liquid engines is constant at 100% and the solid boosters have a simplified two-level thrust profile. Hence, only trajectory optimization variables related to payload mass and pitch and yaw profiles are used, for a total of 11 continuous variables listed in Table 5.17 and already encountered in Table 5.6(B) with the addition of the payload scaling factor. All constraints are used with reference values reported in Table 5.18. An error of 10 km on the final semiaxis, 0.0009 on the eccentricity and 0.5 degree on the inclination are allowed. Trajectory optimizations are performed with vehicle design data fixed to the actual value and with the ones computed through the MDA process.

5.3.3. Numerical Integration. As presented in Section 4.3 direct methods reformulate the infinite dimensional optimal control problem to a finite dimensional NLP problem, discretizing control and state variables.

Constraint	LB	UB
Target semiaxis [km]	$24383.6 - 10.0$	24383.6+10.0
Target eccentricity [-]	$0.7292 - 0.0009$	0.7292+0.0009
Target inclination [deg]	$7 - 0.5$	7+0.5
Maximum axial acceleration [g]	$-\infty$	7.5
Maximum heat flux before payload fairing jettison [MW/m^2]	$-\infty$	40
Maximum dynamic pressure [Pa]	$-\infty$	57000
Controllability violation [-]	$-\infty$	1.5
Maximum angle of attack [deg]	$-\infty$	5

TABLE 5.18. Ariane 5 trajectory constraints bounds.

While stochastic optimizers don't require any assumptions on the model regularity the use of gradient based techniques counts at least on a continuous and differentiable trajectory model. On this purpose different settings have been tested regarding the integration and interpolation methods applied and the redefinition of the optimization problem with multiple shooting techniques to improve also its robustness. In particular two integration and two interpolation techniques are compared:

Integration: adaptive step *Runge-Kutta Fehlberg 4-5*, in its classical formulation and fixed step *Runge Kutta 4*, with an ad hoc procedure for the computation of the sequence of steps.

Interpolation: *linear* univariate and bivariate interpolation, *Akima* spline univariate and *bicubic* bivariate interpolation techniques.

If finite differences are used in the approximation of the derivatives, the use of adaptive step size integration routines can cause discontinuities. On the other hand the use of fixed step size integrators can lead to inaccurate or inefficient integration processes if the step chosen is too large or too small. The solution is to provide a non adaptive integrator such as Runge-Kutta 4 with a sequence of predefined steps. To determine an effective sequence of steps, an initial trajectory integration is performed with Runge-Kutta 4/5 at initialization and at the beginning of every SQP iteration and the determined steps are saved in the memory for the successive QP iterations. This is possible because the phases between the flights of two components are fixed in time. The thrust is at its maximum for all the duration of the flight so the phase duration of each component can be easily computed in advance.
Additionally linear interpolation techniques have been investigated for differentiability issues of the trajectory optimization process. As an alternative Akima spline interpolation for univariate functions has been implemented. The spline between two interpolating points is determined by a third degree polynomial. Four informations are needed to determine this polynomial: the passage through the two points and their slopes, determined locally by the

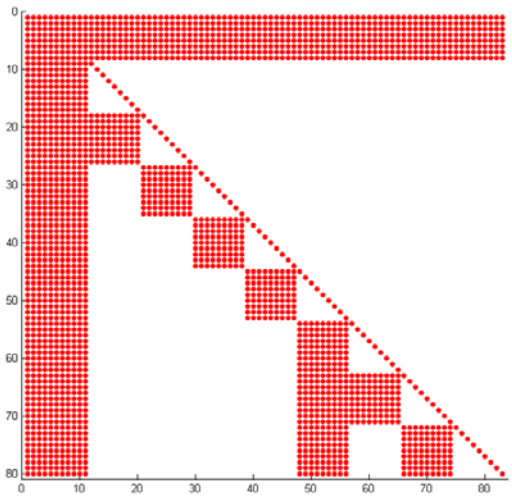


FIGURE 5.27. Example of Jacobian matrix sparsity structure for a multiple shooting definition of the trajectory optimization problem, Ariane 5 to GTO test case.

coordinates of five points in its neighbourhood. For more details about this method refer to [**149**]. A bivariate version of the Akima interpolator is also given in literature [**150**] but for the limited use of the routine in the trajectory optimization problem (mainly used to compute the aerodynamic coefficients) the bicubic interpolation is considered sufficient to solve the differentiability issues.

The optimization result is highly sensitive with respect to the initial values. To improve the robustness of the trajectory optimization process a multiple shooting formulation of the problem has been introduced and its results are compared with the single shooting definition. The idea of the multiple shooting approach, as described in Section 4.3 is that the trajectory is not integrated from the beginning to insertion in the orbit, but on smaller intervals and the joints between the different trajectory segments are guaranteed through continuity constraints. The optimizer shows indeed less sensitivities on the initial conditions. This transcription technique results in a larger and sparser NLP problem, that can be efficiently solved by WORHP, explicitly declaring the structure of the Jacobian and Hessian matrix. Each trajectory segment is detached from the previous one so that each continuity constraint depends just on the states related to the node it applies to and the previous one. For example, the Jacobian matrix reported in Figure 5.27 corresponds to the case of 8 multiple shooting nodes in each component phase. The horizontal band represents the derivatives of the equality and inequality path constraints, their dependency on the complete set of optimization variables is not known a priori. The vertical band refers to the original trajectory variable set of Table 5.17. They directly influence all the trajectory constraints and their sparsity structure is unknown. The rest of the matrix is related to

the additional multiple shooting constraints and variables. In particular for the last phase each continuity constraint in one of the internal nodes depends also on the states at the beginning of the phase due to the intrinsic definition of the bilinear tangent law. Since the Jacobian has dense rows nothing can be said a priori on the sparsity structure of the Hessian matrix. Dense BFGS methods are applied for its approximation. The additional continuity constraints are equality constraints and they are scaled by a factor computed as the ratio between the acceptable error on the state and the optimizer feasibility tolerance in such a way that, denoting by c_i^k the continuity constraints on the k-th node for the i-th state, $c_{\text{norm},i}$ its normalization factor, $\varepsilon_{\text{feas}}$ the feasibility tolerance of the optimizer and $\varepsilon_{\text{state},i}$ the allowed inaccuracy on the matching conditions for the i-th state, it stands that

$$\left|\frac{c_i^k(x)}{c_{\text{norm},i}}\right| < \varepsilon_{\text{feas}} \quad \text{and} \quad |c_i^k(x)| < \varepsilon_{\text{state},i}.$$

It follows that

$$c_{\text{norm},i} = \frac{\varepsilon_{\text{state},i}}{\varepsilon_{\text{feas}}}.$$

Tolerances of 1 km for the position, 1 m/s for the velocity and 0.01 rad for the angles are considered in the results reported below.

5.3.3.1. *Results.* Several trajectory optimization test problems for Ariane 5, with actual and MDA design data, have been run to compare the different settings using WORHP as optimizer

CASE1 - Single shooting, Runge Kutta Fehlberg 4/5, linear interpolation
CASE2 - Single shooting, Runge Kutta 4, Akima and bicubic interpolation
CASE3 - Multiple shooting (1 node for each phase), Runge Kutta Fehlberg 4/5, linear interpolation
CASE4 - Multiple shooting (1 node for each phase), Runge Kutta 4, Akima and bicubic interpolation

The first results reported in Table 5.19 are related to 10 different runs, where the initial guess is provided with a fast global optimization run of the PSO algorithm (10 particles and 10 iterations) to prune the search space and get closer to the global optimum, and the two different design input data sets.
The results are compared in terms of performance, robustness and efficiency. *Performance* refers to the best achieved optimal value, *Robustness* to the standard deviation of the optimal values from the different initial guess runs and *Efficiency* to the mean computational time (maximum allowed computational time 1000 s). Feasibility and Optimality tolerances of WORHP are set to 10^{-4}, acceptable solutions tolerance is 10^{-3} and the sparsity structure of the Jacobian matrix is exploited in the problem implementation.
In the multidisciplinary analysis a fast trajectory optimization with PSO is performed to find a feasible trajectory and define reasonable loads for the structural subsystem. For a fair comparison between the different settings and in order to produce same structural design, the seed given to the PSO

(A) Ariane 5 actual design

	CASE 1	CASE 2	CASE 3	CASE 4
Number of variables	11	11	41	41
Number of constraints	10	10	40	40
(Eq+Ineq)	(3+7)	(3+7)	(33+7)	(33+7)
Optimal solutions	10	9	10	10
Acceptable solutions	0	1	0	0
No solution found	0	0	0	0
Performance [kg]	10165	10177	10167	10179
Robustness [kg]	107.0	2.3	1.6	1.1
Efficiency [s]	287	145	633	828

(B) Ariane 5 MDA design

	CASE 1	CASE 2	CASE 3	CASE 4
Number of variables	11	11	41	41
Number of constraints	10	10	40	40
(Eq + Ineq)	(3+7)	(3+7)	(33+7)	(33+7)
Optimal solutions	8	9	9	10
Acceptable solutions	1	0	0	0
No solution found	1	1	1	0
Performance [kg]	10353	10385	10355	10387
Robustness [kg]	21.9	4.1	1.2	0.5
Efficiency [s]	287	276	620	1102

TABLE 5.19. Ariane 5 to GTO trajectory optimization, initial guess given by PSO, WORHP results.

has been fixed. The guidance laws described in Section 4.2 are used to define a flyable initial guess solution obtained with the midpoint values of the control variables box constraints. The results obtained by this initial guess are reported in Table 5.20 for the actual vehicle design and for the design given by the multidisciplinary analysis.
The results are compared with the one obtained with the stochastic PSO strategy. For the two test cases, actual and MDA design, 5 runs of PSO have been executed to evaluate the algorithm capability and the entity of the stochastic effect. Several swarm sizes have been tried maintaining constant number of function evaluations, and better results are obtained with 250 particles. The PSO parameters are the default ones and the maximum number of iterations is set to 500. The results are reported in Table 5.21. *Performance* refers to the maximum value of payload mass obtained in the different runs, *Robustness* to the standard deviation of the optimal value from

(A) Ariane 5 actual design

	CASE 1	CASE 2	CASE 3	CASE 4
Solution	Optimal	Optimal	Optimal	Optimal
Performance [kg]	10164	10169	10165	10175
Efficiency [s]	194	198	970	959

(B) Ariane 5 MDA design

	CASE 1	CASE 2	CASE 3	CASE 4
Solution	Optimal	Optimal	Optimal	Optimal
Performance [kg]	10353	10383	10354	10386
Efficiency [s]	383	811	491	851

TABLE 5.20. Ariane 5 to GTO trajectory optimization, initial guess by standard guidance laws, WORHP results.

	Actual Design	MDA design
Performance [kg]	10157	10365
Robustness [kg]	245.9	209.8
Efficiency [h]	3.9	4.1

TABLE 5.21. Ariane 5 to GTO trajectory optimization results, random seed generator, PSO optimizer, actual and MDA design.

the different runs with different seeds and *Efficiency* is the mean value of the computational time required in each run.

5.3.3.2. *Conclusions.* The ascent trajectory optimization problem is solved using local deterministic and global stochastic optimization techniques. Particular attention has been paid for assessing the robustness of the local approach, based on the evaluation of the gradient of objective and constraints, solving the arose issues related to the smoothness and differentiability of the model. Moreover, to reduce the sensitivity of the optimal control problem to the initial conditions of the system, the transcription of the optimal control problem into an NLP problem using multiple shooting techniques have been also included in the analysis.

WORHP is able to overtake in terms of efficiency, robustness and performance the evolutionary approach. Especially the use of direct transcription methods allows to obtain convergence to the optimal solution, in all the test runs starting from different initial guesses, with a payload deviation of the order of 1 kg and the best achieved performance value. The advantage in terms of payload performance and robustness of the optimization process is paid by an increase in the computational efficiency. This is due to the augment of the optimization variables and constraints introduced by the multiple shooting definition of the problem. If also the path constraints are discretized over

the time the optimization problem will considerably enlarge but the sparse block structure of the Jacobian matrix will allow the use of more efficient sparse BFGS techniques. The good results obtained with the local approach are strictly related also to the definition of the optimization problem. The engineering expertise in defining the search space control variables is crucial for the choice of the initial guess and the convergence of the algorithm.
The goal of efficiently optimizing the rocket payload performance was twofold: running stand-alone trajectory optimization for existing vehicles freezing the design to their actual value and running nested trajectory optimization in the multidisciplinary vehicle analysis. WORHP was the best optimization approach in both cases. The use of transcription methods are foreseen for the improvements brought to the robustness of the optimization process. While computational efficiency is not an issue when the trajectory is optimized independently from an MDO problem, it is crucial when it is nested in every MDO analysis evaluation. Based on the achieved results, the single shooting definition of the optimal control problem is the computationally most efficient approach, even if lacking in robustness and success in convergence. The robustness of the optimization process using single shooting technique can be improved using non linear interpolation and fixed step integrators.

5.3.4. Semianalytic Integration. The semianalytic method presented in Section 4.4 has been validated in terms of integration accuracy and optimization efficiency. The basic idea of the method is to find analytic solutions of the equations of motion on small intervals by means of Taylor expansions and wise manipulations. The improvements are mostly in term of efficiency. The number of subintervals where the analytic solution is calculated and the number of evaluations of the system of differential equations is reduced compared to the number of intervals needed by the numeric integration.
The integration of the ascent trajectory with semianalytic methods depends on 4 parameters, related to the accuracies for the computation of the step size. In Table 5.22 the results obtained for different set of tolerances are summarized. The test is performed on the initial guess trajectory of the problem described in Table 5.17, for Ariane 5 launcher with frozen design. The results are compared with the integration obtained with Runge Kutta 4/5, for different accuracies on the relative error, in terms of number of steps computed (n_{step}), number of evaluations of the EoM (n_{eom}), norm of the error on the final states scaled by the numeric reference value computed with $\varepsilon_{\text{rel}} = 10^{-8}$, and computational time.
The results comply the expectations: the semianalytic method is able to reduce the time required for the trajectory integration, decreasing the number of steps and the calls to the system of differential equations. The efficiency of the analytic approach is inversely proportional to its accuracy. The best compromise between accuracy and efficiency is represented by the setting $\varepsilon_{acc} = \varepsilon_{\gamma} = 10^{-7}$ and $\varepsilon_r = \varepsilon_v = 10^{-1}$ with an error of 0.5% on the final states. The tolerances on the exoatmospheric part are kept larger compared

(A) Numerical integration

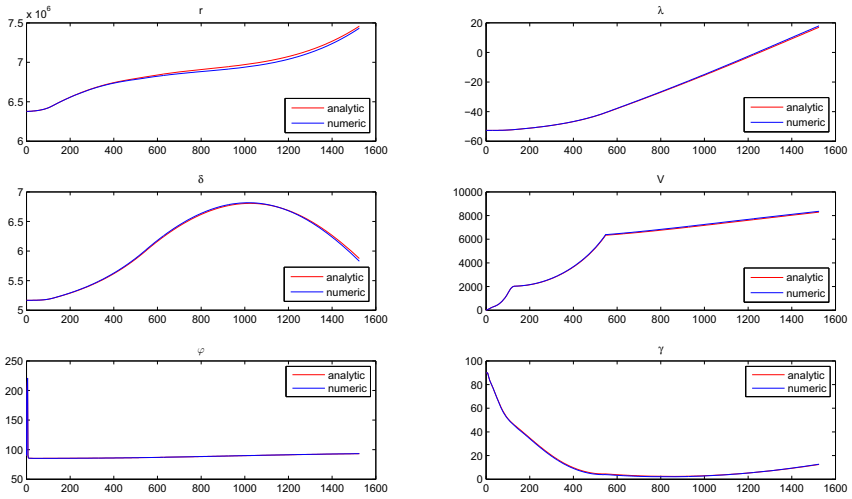
ε_{rel}	10^{-8}	10^{-7}	10^{-6}	10^{-5}
n_{step}	1789	1669	1631	1602
n_{eom}	11176	10393	10065	9786
err	0.0	0.0001	0.0008	0.0028
CPU [s]	0.118	0.110	0.107	0.105

(B) Semianalytic integration

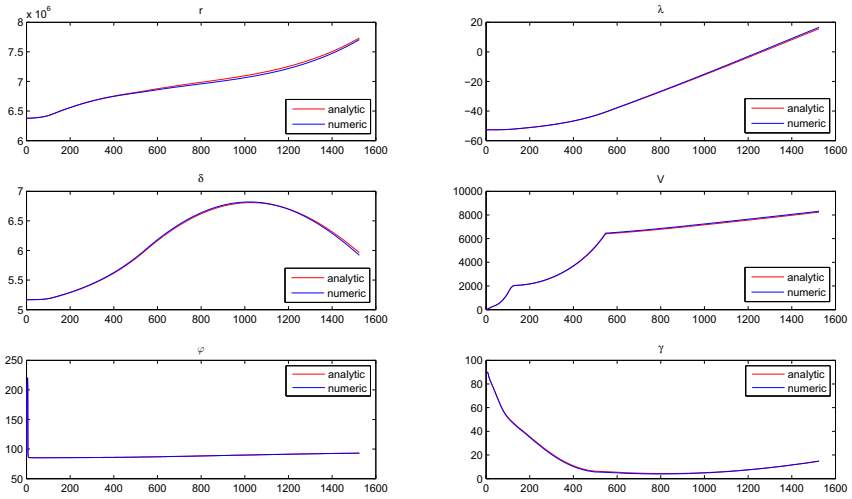
$\varepsilon_{\text{acc}}, \varepsilon_\gamma$ $\varepsilon_r, \varepsilon_v$	10^{-8} 10^{-1}	10^{-7} 10^{-1}	10^{-6} 10^{-1}	10^{-5} 10^{-1}
n_{step}	1433	943	715	605
n_{eom}	2708	2218	1990	1880
err	0.0002	0.0005	0.0058	0.0169
CPU [s]	0.075	0.069	0.066	0.064

TABLE 5.22. Ariane 5 to GTO trajectory integration. Semianalytic and numeric efficiency results comparison.

to the atmospheric ones. This is because the exoatmospheric part of the flight is less sensitive on perturbations of its states. The results are related to the actual design case. An analogous study can be performed also for the multidisciplinary design analysis. Since the design, either the actual one or the MDA one, is kept fixed when the simulation of the trajectory is performed, an analogous study for the MDA case would not bring additional information. However, in Figure 5.28 the initial guess trajectories used in the following results obtained through the simulation of the trajectory with the numeric and semianalytic approach for Ariane 5 actual and MDA design are plotted. The initial guess is obtained with the optimizable control parameters equal to the mid point of the box constraints. The plots refer to the state variables values, in the local coordinate system, with respect to the time of flight.
The same tolerances are used to solve trajectory optimization problems comparing the single shooting settings of CASE 1 and CASE 2 presented in Section 5.3.3, where the second case reduces just to the application of non linear interpolation. As for the numerical case 10 different optimization runs with 10 different initial guesses given by PSO are performed and the results are compared in terms of performance, robustness and efficiency. The results are collected in Table 5.23. They must be interpreted in relation with the one obtained for the numerical case. For a fair comparison, in case of a design given by the multidisciplinary analysis, the PSO seed for assessing the trajectory load is fixed to the same values. In all the considered test cases the vehicle performance is in average 50 kg less overestimated than in the



(A) Ariane 5 actual design



(B) Ariane 5 MDA design

FIGURE 5.28. Integration of the initial guess trajectory. Comparison between semianalytic and numerical methods.

results obtained with the Runge-Kutta integrator and 50 kg closer to the Ariane 5 reference value. This is due to the small error on the simulation of the trajectory introduced by the use of the semianalytic approach.
The main advantage is definitely in terms of computational efficiency. The average computational time required for optimization reduces to almost one third in CASE 1 and halves for CASE 2, just for the case of MDA design, with respect to the numerical case. As already observed, the use of non linear interpolation increases the optimization robustness affecting in this case

	Actual design		MDA design	
	CASE 1	CASE 2	CASE 1	CASE 2
Number of variables	11	11	11	11
Number of constraints	10	10	10	10
(Eq+Ineq)	(3+7)	(3+7)	(3+7)	(3+7)
Optimal solutions	9	9	10	9
Acceptable solutions	1	0	0	0
No solution found	0	1	0	1
Performance [kg]	10105	10118	10296	10329
Robustness [kg]	12.2	6.1	5.8	4.5
Efficiency [s]	87	179	97	165

TABLE 5.23. Ariane 5 to GTO trajectory optimization with semianalytic integration, initial guess given by PSO, WORHP results.

the computational time. The single shooting definition of the optimization problem and the use of linear interpolation is foreseen for nesting the trajectory optimization loop within the multidisciplinary analysis. In this way each evaluation of the model requires approximately 1 to 2 minutes.

5.4. Comparison of MDO Architectures

The global MINLP optimization algorithms have been validated in Section 5.1 and 5.2 against analytic and MDO test problems, using the BBO architecture and the MDF problem formulation. Furthermore stand-alone trajectory optimization for fixed vehicle designs has been verified for accomplishing the best compromise between efficiency, performance and robustness. Consequently to the performed studies, the enhancements to the MDO architecture and the problem formulation are analyzed here.

5.4.1. MDF vs IDF. As already outlined in Section 2.4 the MDO problem of ELV can be either defined as MDF or partial IDF. The partial IDF formulation doesn't guarantee multidisciplinary feasibility between outputs and inputs of the trajectory and structural subsystems. In the current version of the MDA, targeting ELV at an early preliminary level of detail, there is an internal loop between the trajectory and the structure blocks. The trajectory analysis gives as output the load values to the structural analysis to dimension the inert masses of stages, boosters and payload fairing according to these values and return structural values to the trajectory subsystem in an iterative loop that achieves convergence if a feasible trajectory is reached. In the partial IDF formulation the internal cycle is broken and additional design variables are added to the MDO problem. These variables are modeling the inert masses and they are used as input to the trajectory analysis. Additional constraints for matching the trajectory input with the output coming from

the structural analysis are added to the optimization problem.
The comparison between the two different MDO problem formulations are performed for the "small" Ariane 5 MDO problem. The simplicity of the problem will allow an easy comparison. The MDO problem with MDF and partial IDF formulation is solved for single objective optimization with the global PSO algorithm and since there are no integer variables the solver WORHP is used. Three runs of PSO with different initial populations are considered with the default set of parameters, 500 particles and 200 maximum number of iterations. A deviation of $\pm 30\%$ from the inert masses computed by the weights module is allowed for the introduced optimization variables. The WORHP algorithm is run with initial guess equal to the midpoints of the box constraints which represents also the actual Ariane 5 design configuration.
The results are compared in Figure 5.29. The minimization of the total mass at launch is the optimization objective. WORHP exits when the feasibility and optimality conditions are met with a tolerance of 10^{-4}. For a fair comparison, the achieved objective values are plotted against the normalized CPU time required for convergence.
In numerous examples taken from the literature (see references in Section 2.3.8) the IDF formulation was shown to provide significant advantages in terms of computational efficiency. The main advantage of removing the internal loops is the reduction of the computational time needed for the single model evaluation. Considering that, multidisciplinary analysis with long iteration cycles and hence long evaluations will gather good advantages from the IDF approach.
In the considered MDO problem the ratio between the execution of the model evaluation with the MDF and IDF formulation for the Ariane case is in average equal to 1.97 as stated in the parallel research work in reference [**52**]. Usually 2 or 3 iterations between the trajectory and structure subsystems are enough for achieving multidisciplinary feasibility with a tolerance of 1% on the inert masses. For this reason the noticeable saving in computational time found in literature for the IDF approach is not experienced here. For the PSO case, at equal amount of allocated computational resources, the MDF formulation is bringing a considerable advantage in the final objective value. This is attributed to the increase of complexity of the partial IDF problem formulation which enlarges the number of design variables and constraints. Hence the speed-up gained with the partial IDF formulation is not enough to compensate the growth of the problem complexity.
For the deterministic approach, on the contrary, the enlargement of the optimization problem gives more freedom to the algorithm of exploring multidisciplinary unfeasible areas of the search space and converging to an improved optimal value if compared with the MDF definition. The advantage in performance is paid by the computational time required for convergence. The convergence to the optimum with the partial IDF approach is accomplished in greater time than with the MDF formulation, confirming what stated before,

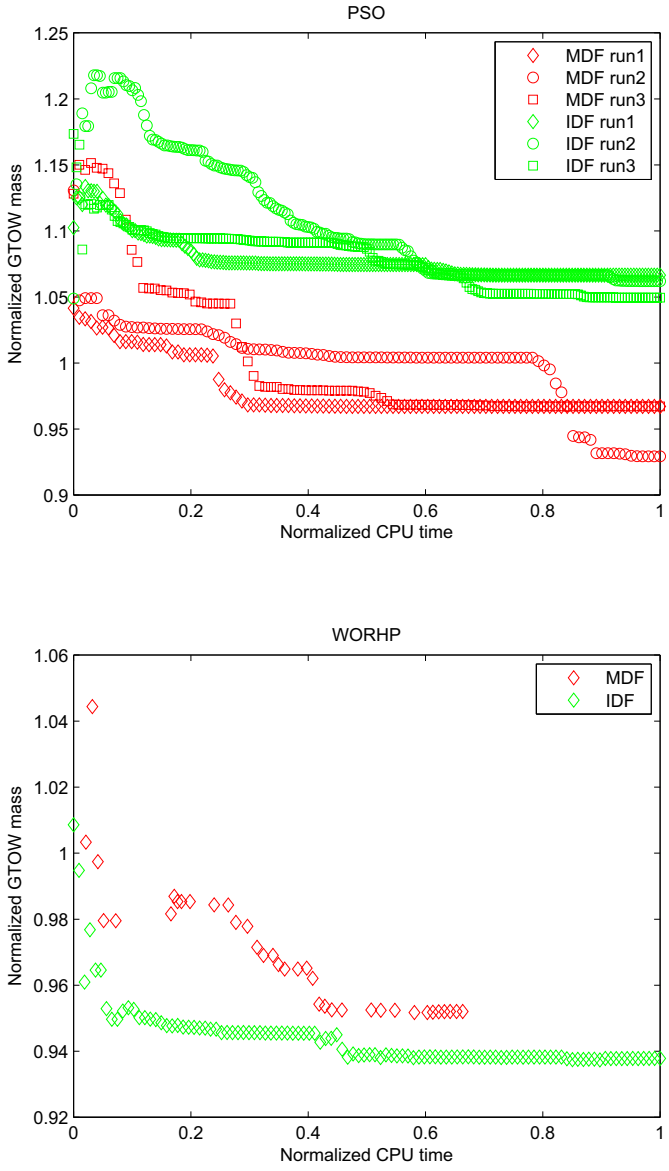


FIGURE 5.29. Comparison of MDF and IDF problem formulation for the small MDO problem of Ariane 5 for global stochastic and local deterministic optimization techniques.

that the speed-up gained in the model evaluation is not enough for a faster convergence of the optimization process.

5.4.2. BBO vs NOL. As introduced in Section 2.4, two MDO architectures have been compared: BBO and NOL. The first one, that is also the one used in the previous examples, is structured with just a global optimizer on top of the MDA. The second one, instead of performing a simulation of the

		200 particles		500 particles	
		BBO	**NOL**	**BBO**	**NOL**
Seed 1	GTOW [tons]	-	742.0	766.4	748.8
	CPU [h]	11.3	5.3	1.4	3.9
Seed 2	GTOW [tons]	765.9	754.4	765.4	725.4
	CPU [h]	2.7	1.8	2.2	9.3
Seed 3	GTOW [tons]	-	744.0	766.1	753.0
	CPU [h]	11.1	7.6	3.4	12.1

TABLE 5.24. BBO and NOL MDO architecture comparison.

trajectory inside the evaluation of the multidisciplinary analysis, performs a fast trajectory optimization using semianalytic methods at each evaluation of the MDA.
To compare the two architectures the internal cycle between the trajectory and the structure subsystems has been broken. The input loads to the trajectory blocks are the one returned by the weights subsystem and the structure is computed with the results coming from the optimization or simulation of the trajectory.
For only test purpose the simple single objective optimization problem of Section 5.1.2 is used, without the possibility of optimizing the main structural material of the boosters set. The equality trajectory constraints are further relaxed admitting an error of 500 km on the target semiaxis, 0.002 on the target eccentricity and 0.5 degrees on the target inclination. The PSO algorithm is used as global optimizer with its default set of parameters. As the objective is the minimization of the GTOW the trajectory optimization subproblems have been performed with constant objective, the payload mass, hence just feasibility is achieved for convergence. To avoid waste of computational sources the trajectory optimization in the NOL architecture is performed just on feasible design solutions, otherwise a simple simulation is computed and the evaluation moves to the next swarm particle.
For a fair comparison between the two architectures the MDO is stopped whenever a feasible solution that improves the current value of the total mass at launch of Ariane 5 is found.
Since the check of the stopping condition is performed at each swarm evaluation, to return the best feasible solution found, two different set of swarms are used: a small swarm formed by 200 particles to advantage the NOL architecture that takes longer time to perform a single iteration, and a bigger swarm formed by 500 particles to advantage the BBO architecture in exploring the search space. The maximum number of iterations for the stochastic algorithm is fixed equal to 400 and the nested trajectory optimization has set a maximum time of 2 minutes. Three different swarm initializations are considered for stochastic effect. The results are summarized in Table 5.24.

The NOL architecture is achieving the best objective values in both configurations. This is due to the advantage brought by the solution of the trajectory optimization subproblem, performed for each feasible design configuration, that is able to converge to trajectory-feasible solution even if its initial guess has a high constraint violation. The main obstacle for the BBO architecture was the fulfillment of the constraints on the final orbit. If the trajectory constraints are treated at the same level as the other constraints the stochastic algorithm has limited capabilities in converging to a feasible solution. Increasing the size of the swarm gives to the PSO and to the BBO architecture the advantage of having a wider insight over the design search space and increase the chances of placing a particle in a trajectory-feasible region. With a NOL approach there is the advantage of treating the hard trajectory constraints separately with efficient optimization techniques. The main drawback is the computational time. If the size of the swarm enlarges the single swarm evaluation can take hours. It must be notice that in the results reported here the NOL architecture was able to find improved GTOW values in maximum 2 iterations.

5.4.3. Local Refinements. As introduced in Section 2.4, the solutions returned by the global optimizer can be further refined using local gradient based optimization techniques. In this way the strengths of the two optimization approaches are exploited: the wide exploration capabilities of the global algorithm to prune the search space and dealing with discrete variables and the efficiency of the local one to exactly converge to the nearby optimum.
The example reported here refers to the "large" MDO problem of Ariane 5, already presented in Section 5.2.2.2. The optimization objective is the minimization of the GTOW and the global optimization is performed with the stochastic PSO algorithm for 100000 function evaluations. The solution returned by the PSO is further refined by WORHP. Since no discrete variables are involved in the MDO the complete set of continuous design parameters is used also in the refinement. In case of discrete variables those must be fixed to the value returned by the global algorithm.
The results are reported in Table 5.25. The design values obtained running the MDA on the inputs given by the actual Ariane 5 design are compared with the global optimum found by the PSO algorithm and its local refinement with WORHP.
The local refinement is bringing further improvements to the solution returned by the PSO algorithm. The large bounds on the optimization variables enlarge the design search space, and the global optimization approach for the given number of function evaluations is not able to converge to the global optimum, even if narrowing the design search area that contains it. However the local strategy can though efficiently converge to an improved solution, in the neighborhood of its initial guess, reducing by additional 4.3% the total mass at launch.
The local refinements of global multi-objective MDO problems are performed

Variable	MDA	PSO	WORHP
$(L/D)_{\text{PLF}}$	1.03	1.38 (+34.4%)	1.43 (+39.2%)
$M_{\text{prop,EPC}}$	173.3	152.5 (-12.0%)	146.6 (-15.4%)
T_{EPC}	1293.3	1283.1 (-0.8%)	1224.9 (-5.3%)
$\alpha_{\text{P,EPC}}$	6.1	6.6 (+7.9%)	6.3 (+3.3%)
ε_{EPC}	59.5	45.2 (-24.0%)	46.5 (-21.8%)
$p_{\text{cc,EPC}}$	115.0	104.8 (-8.9%)	109.8 (-4.5%)
$p_{\text{tanks,EPC}}$	3.0	2.2 (-26.7%)	2.2 (-26.7%)
$p_{\text{press,EPC}}$	23.0	29.9 (+30%)	29.9 (+30%)
$M_{\text{prop,ESC-A}}$	14.4	16.5 (+14.6%)	16.8 (+16.7%)
$T_{\text{ESC-A}}$	63.0	69.4 (+10.2%)	82.3 (+30.6%)
$\alpha_{\text{P,ESC-A}}$	5.0	4.9 (-2.0%)	4.9 (-2.0%)
$\varepsilon_{\text{ESC-A}}$	83.1	81.3 (-2.2%)	83.8 (+0.8%)
$p_{\text{cc,ESC-A}}$	37.0	48.5 (+31.1%)	50 (+35.1%)
$p_{\text{tanks,ESC-A}}$	3.0	2.9 (-3.3%)	2.9 (-3.3%)
$p_{\text{press,ESC-A}}$	226.0	268.7 (+18.9%)	278.2 (+23.1%)
$M_{\text{prop,P241}}$	240.1	201.4 (-16.1%)	189.4 (-21.1%)
D_{P241}	3.05	2.51 (-17.7%)	2.44 (-20.0%)
T_{P241}	6395.0	5096.7 (-20.3%)	4874.3 (-23.8%)
$\varepsilon_{\text{P241}}$	11.0	8.5 (-22.7%)	8.4 (-23.6%)
$p_{\text{cc,P241}}$	64.0	63.1 (-1.4%)	61.6 (-3.8%)
GTOW [tons]	**770.8**	**658.4 (-14.6%)**	**625.2 (-18.9%)**

TABLE 5.25. Ariane 5 ECA MDO, large bounds, minimization of the GTOW mass. PSO solution and WORHP local refinement.

selecting a solution from the non dominated front and refining it with respect to one of the objectives or a linear combination of them.

5.4.4. Conclusions. Several improvements to the BBO architecture with MDF formulation have been tested. The IDF formulation of the MDO problem, even if it halves the evaluation time of the cost function, is not bringing enough computational saving for recording advantages in the convergence capability of the optimization algorithms. Even though it must be noticed that in the given example a deterministic optimization approach with an IDF formulation is able to converge to an improved optimal solution. The enlargement of the problems in terms of number of design variables and constraints is preventing the stochastic algorithm from convergence while favoring the exploration capabilities of the deterministic approach.
The comparison of the BBO and NOL architectures generates contrasting results. The NOL approach overtakes the BBO in terms of quality of the

solutions returned while the latter one has the best computational savings. With the possibility of parallelize the global optimization algorithm the NOL architecture is expected to be the most promising approach. At the current state, since just an OpenMP parallelization is available, the BBO architecture is the best compromise achieved between quality and efficiency.
The combination of global and local search techniques is revealed to be a further enhancement of the proposed architecture. The local refinement of the solution returned by the global algorithm is contributing to the improvement of the approximation of the global optimum. However the effectiveness of a local optimization is related to the global search and to the problem dimension. In MDO problems where the design is too much constrained the global algorithm itself is capable of achieving convergence in a suitable number of function evaluations. A local refinement will not bring prominent upgrades. Instead, for larger design search spaces, the collaboration of global and local searches can bring a positive contribution to the final objective value at the price of a small amount of additional computational time.

CHAPTER 6

Conclusions and Future Developments

MDO is still a new design methodology that needs to consolidate and be accepted by the industrial community as an alternative to the traditional fixed point iterations design approach. The present research activity brought a contribution in the statement of MDO as effective technique for achieving sound design modifications through optimization, when applied to the design of Expendable Launch Vehicles. Moreover, this work enlarges the range of optimization techniques that can be used in an MDO framework, developing ad hoc methodologies and a suitable definition of the optimization problem and MDO architectures thus also drawing some important guidelines from the comparison of the different approaches. In particular, the main worth mentioning accomplishments can be summarized as follows:

- Literature research on available MDO techniques: novel classification of the available MDO architectures based on the optimization problem formulation (IDF-MDF-AAO) and the analysis decomposition (ND-HD-NHD).
- Global and local optimization techniques survey for solving MINLP problems with integer and categorical variables. Distinction between deterministic and stochastic algorithms for solving single and multi-objective optimization problems.
- Comparison between deterministic and stochastic optimization approaches on a set of analytic and applicative MDO problems for single and multiple objectives.
- Internal development of a hybrid stochastic strategy for multi objective optimization and a combined branch and bound and gradient based technique for deterministically solving single objective problems, with the possibility of reformulation of the model functions for tackling categorical variables.
- Enrichment of the stochastic strategies with stopping criteria which are not based on an a prior knowledge about the optimal solutions set and parallelization of the algorithms using OpenMP paradigm to improve the computational efficiency.
- Resolution of the stand alone ascent trajectory optimization problem, using numerical and semianalytic integration techniques and direct transcription methods for optimal control. Resolution of the optimization problem with both deterministic NLP techniques and stochastic algorithms.

- Comparison of suitable MDO architectures and problem formulations for the multidisciplinary design optimization of ELV: BBO and NOL architectures with further local refinements of the global optimal solution, MDF and IDF problem formulations.

Considering the presented study, it is possible to draw the following most important conclusions

- For single objective optimization the deterministic BBWORHP technique resulted to be largely more efficient than the stochastic PSO strategies for solving analytic and MDO single objective optimization problems. However its applicability strictly relies on the quality of the first guess and on the number of optimization variables involved. The possibility of solving MINLP problem with categorical variables, hence not relaxable, is a further enhancement for the applicability of the branch and bound algorithm to MDO problems which, to the knowledge of the author have not yet been tackled in former studies.
- For multi objective optimization, the stochastic evolutionary approach compared with the derivative free algorithm MADS resulted to be the most effective and efficient methodology. In particular the self developed hybrid strategy, combining three evolutionary algorithms from the class of ant colony, genetic algorithm and particle swarm optimization in a cooperative optimization loop, has been proved to be able to achieve the expected results. On both the analytic and the MDO problems the HGO algorithm was outperforming each of the hybridized evolutionary strategies in terms of efficiency and quality of the solution returned. Efficiency is still an issue for stochastic algorithms, which can be partially solved by an OpenMP parallelization for shared memory machines implemented.
- The possibility of further refining the solution returned by the global MDO using efficient gradient based techniques for NLP problems can bring substantial improvements with a small allocation of computational resources. However, the worthiness of applying a local refinement depends on the dimension of the search space of the optimization problem and on the performance of the global algorithm.
- The ascent trajectory optimal control problem has been solved in the most robust way by using the multiple shooting transcription of the problem and local, gradient based, optimization techniques. Besides the introduction of the semianalytic integration of the trajectory allowed to obtain good results reducing of almost one third the computational time for optimization required by the numeric approach.
- For the given MDO problem the partial IDF formulation is not giving the same advantages outlined in literature. This is due to the inexpensive internal cycle between the trajectory and structure subsystems that doesn't allow to exploit at maximum the power of an IDF approach. The MDF formulation is still the most effective.

- The BBO architecture is the best compromise between optimization process efficiency and effectiveness. However the NOL approach, making use of semianalytic integration methods, achieved promising results in terms of convergence. Inefficiency is still a drawback that can be solved in the future with a massive parallelization.

The main weak point of the proposed approach and some ideas for future developments conclude the dissertation

- The proposed optimization strategies are able to converge to the optimal solutions but the required computational resources are still demanding. Some form of parallelization is needed for the stochastic algorithms and for extending the applicability of the branch and bound technique to higher numbers of integer variables. The parallelization through MPI, for distributed memory machines, can prominently shorten the computational time required for optimization.
- Deterministic techniques are generally more efficient than the stochastic ones. New promising optimization techniques, such as the readaptation of the SQP methods for MINLP problems proposed in [**151**], are foreseen in the future for application to the MDO problem. On the other hand the proposed deterministic techniques are only locally convergent. A global-local hybrid procedure for the pruning of the search space and the successive local search close to the global minimum, is necessary.
- The possibility of introducing external tools and higher fidelity models is the next step of the MDO analysis for ELVs presented here. If the evaluation of the vehicle analysis becomes too expensive, the possibility of including metamodeling techniques such as response surface methodologies needs to be considered. The actual model is hence evaluated in some points, selected for example with the design of experiments approach, and the derived surface that approximates the model can be used for the optimization process.

MDO is expecting to have a great impact in the future design methodologies. However its development and the real employment of MDO in an industrial contest must be growing in parallel with the application of high performance computing techniques and exploitation of new hardware technologies. Its applicability is not restricted to the aerospace industry but can be expanded to any kind of system design that presents the coupling of two or more disciplines. However, it has been shown in this thesis work that, even at the current state of the art, MDO environments already represent a powerful tool for the initial design phases of space transportation systems, when properly guided by human expertise.

APPENDIX A

Commercial MDO Tools

Software	Main Features
AML, AMOpt [**152**], Technosoft, 2002	Interfaces with existing tools for structural analysis and post processes analysis. Generative modeling. Integration of third party applications. XML data handling. Process Parallelization. Visualization tools. Multiplatform.
BOSSQuattro [**153**], Samtech, 1997	Open design and optimization architecture for parametric analyses, design of experiments, multidisciplinary optimization and sensitivity analysis, statistic analyses and updating. It can make use of internal solvers or integrate external optimization algorithms.
Caffe [**154**], Desktop Aeronautics, 2000	Collaborative Optimization framework. Integration of existing code for analysis and optimization. Management of the design process on multiple distributed platforms. GUI. XML data handling.
DAKOTA [**155**], Sandia Web, 2009	Flexible and extensible interface between simulation codes and analysis methods. Containing algorithm for deterministic and stochastic optimization, parameter estimation and sensitivity analysis. Multilevel parallel object oriented framework.
DesignXplorer [**156**], ANSYS, 2009	Analysis of parametric designs with response surfaces and design of experiments methodologies. Visualization tools. Multiobjective optimization algorithms.
FASTPASS [**157**], Lockheed Martin Astronautics, 1996	Optimization of launch vehicle design and components detailed design. Not extensible to external models.
FIDO [**158**], NASA, 1993	Massively parallel computer architecture. Disciplinary codes run on different processors. GUI. Optimization routines. Flexible and modular architecture that can adapt to different MDO applications.
HEEDS MDO [**159**], Red Cedar Technologies, 2004	Interface only with a list of commercial Computer Aided Engineering (CAE) tools (such as Nastran, Excel, ANSYS WB...). Single-objective and multi-objective optimization. Design of Experiments and Reliability analysis. GUI.

HyperWorks - HyperStudy [**160**], Altair Engineering, 1999	Design of experiments. Meta modeling approximations. Collection of single and multiobjective algorithms. Stochastic studies. Post processing and Data Mining. Parameterization of analysis models.
IMAGE - DREAMS [**161**], Georgia Tech, 1996	GUI driven and Internet enabled design framework. Customized framework with the Agent Integration module for custom analysis tools. UNIX operating system.
IOSO [**162**], Sigma Technology, 2008	Compatible with almost all CAM/CAD/CAE applications both commercial and in-house. MDO approach based on the simultaneous use of multilevel, multicriteria and parallel optimization. Optimization performed with a novel evolutionary response surface strategy.
iSIGHT [**163**], Dassault Systèmes Simulia, 2007	Capability of include commercial CAD/CAE software and internally developed programs. Interfaces for custom applications and Excel spreadsheets. Design of Experiments, Optimization, and Approximations technologies.
LS-OPT [**164**], Livermore Software Technology Corporation, 1995	Custom objective and constraints definition. Design exploration with meta-modeling, response surfaces methodologies. Sensitivity and robustness analysis. Parameter identification.
Kimeme [**165**], Cyber Dyne, 2011	Platform for multi-objective and multidisciplinary design optimization. Coupled, by means of scripts, with third-party software. Integration of custom optimization and/or analysis algorithms. Graphical design environment for problem definition, analysis and visualization of the results. Software network infrastructure to distribute the computational load.
MDICE [**166**], NASA, 1998	Multidisciplinary Analysis. Interface with commercial software for computer aided design, grid generation, computational fluid dynamics, computational structural dynamics. Visualization tools. Computing environment for the concurrently and cooperatively operation of many computers.
ModelCenter [**167**], PhoenixIntegration, 1998	Visual environment. Workflow graphically constructed. Data Fitting. Quick wrapping of batch mode programs into the modeling environment. Up to 30 optimization algorithms with definition of objectives, variables and constraints.

modeFRONTIER [**168**], ESTECO 1998	Multi-disciplinary and multi-objective optimization and design environment. Coupling to many existing computer aided engineering tools. Post processing results analysis. Visual environment. Simultaneous use of simulation software on different machines.
Nexus [**169**], iChrome, 2011	Linking to a list of third party commercial tools. Plug-ins for specific custom analysis tools. Trade-off, design of experiments, statistical analyses, response surface and metamodelling studies. Multi-objective optimisation algorithms. Visual environment.
OptiY [**170**], OptiY e.K., 2005	Multidisciplinary design environment. Providing direct and generic interfaces to many CAD/CAE systems, intern codes and externs programs through predefined interfaces. Graphical workflow editor. Modern optimization strategies, probabilistic algorithms for uncertainty, reliability, robustness, sensitivity analysis, data-mining and meta-modeling.
OPTIMUS [**171**], Noesis Solutions, 1996	Process Integration and Design Optimization software. Design of experiments and response surface modeling for design space exploration. Visual environment. Graphic workflow editor.
PASS [**172**], Desktop Aeronautics, 2005	Applicable to Aircraft Design. Rapid analysis coupled with optimization tools. Wide range of appropriate, real-world constraints. It is built on a modular, extensible framework that allows for the implementation of higher-fidelity analysis codes into the conceptual design process. Visual environment.
SmartDO [**173**], FEA-Opt Technology 1992	Collection of deterministic and stochastic optimization strategies for concurrent sizing, shaping and topology design optimization. Integrate of CAD/CAE tool. Processing external data for further parametrization analysis and optimization. Visual environment.
VisualDOC[**174**], Vanderplaats Research and Development, 1998	Multidisciplinary design, optimization, and process integration software. Optimization, design of experiments, response surface approximation, and probabilistic (robust and reliability-based) analysis. Integration of virtually any CAE analysis software. Graphic workflow editor.

TABLE A.1. Survey of existing commercial and academic MDO tools

APPENDIX B

SVAGO Software Tool

The goal of the research was set as the development and quantitative assessment of an MDO software environment capable of tackling the early design phases of ELVs, which was named Space Vehicle Analysis and Global Optimization (SVAGO). ELVs were selected as the primary application due to the current European relevance (Ariane 5 ME, VEGA evolutions, Future Launchers Preparatory Program, ...), but SVAGO was developed from the beginning with a strong consideration for possible future extensions to other classes of Space Transportation Systems (STS).
In particular, the optimization architecture was designed to be completely problem independent, for an easy future maintainability and extendibility of the tool, and a preliminary design of the engineering models for reentry vehicles and reusable launchers was drawn. The development of the software was performed step by step, targeting first a conceptual level of detail for the first release of the software (June 2011) and then the early preliminary level for the second version (March 2012).
The software is written in C++ object oriented language, with parallel development and testing on both Windows (MSVC 2008) and Linux (gcc 4.4) platforms. SVAGO can be executed in bash mode or through an easy-to-use graphical interface designed using Qt 4.6. The main input/output and data storage is achieved through XML files, whereas parallelization of the stochastic global algorithms for shared memory machines is achieved under the OpenMP paradigm. The core of the MDO environment is constituted by 7 software elements:

(1) The Graphical User Interface (GUI) manages the communication between the user and the internal routines of the software. Through the GUI, the user can define all inputs required for the execution of any type of analysis or optimization within SVAGO. The GUI then automatically generates an XML file which constitutes the input of SVAGO's execution. During such execution, status messages are reported within the GUI's console, and at the end of the analysis or optimization the geometry of the vehicle being designed is shown in the GUI's geometry viewer (Figure B.1). After any run, the user can save all data regarding the vehicle stored in the software memory to an XML output file, which has the same structure as the input file. If multiple vehicles are obtained (i.e. multi-objective Pareto fronts), a separated XML output file is saved for each design solution.

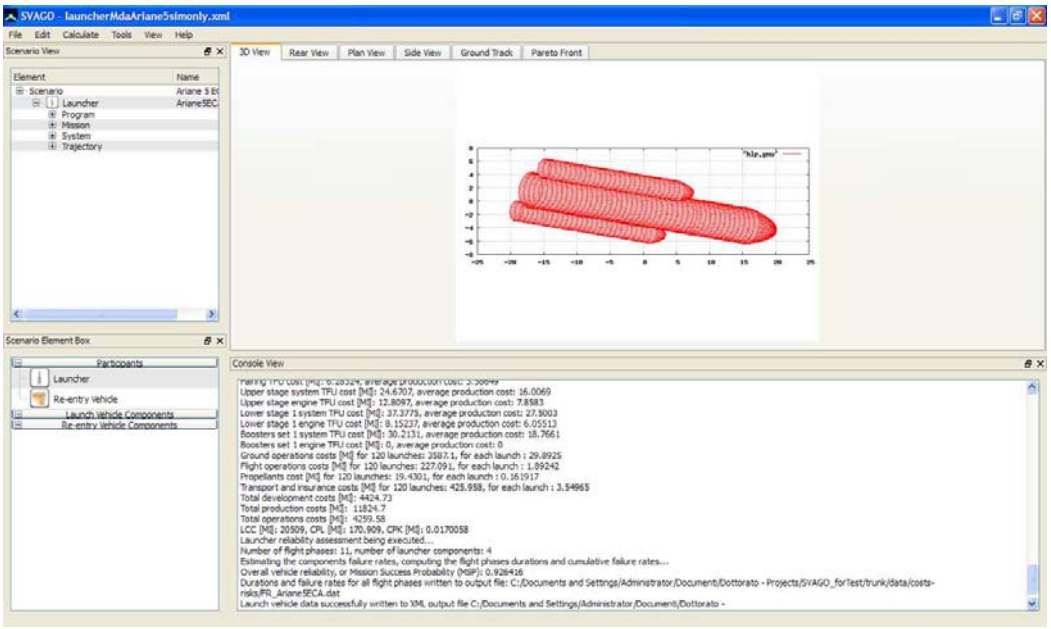


FIGURE B.1. Screenshot of SVAGO software tool in Windows XP.

(2) The Central System Database (CSD) is a structure in C++ which matches the XML input/output file structure and therefore constitutes the main database of the software tool. Initially, all information from the XML input file is automatically stored into the CSD, including all user selections, mission parameters, vehicle design constraints and optimization settings. As an analysis proceeds, output data from each discipline are also stored in the CSD as well as all optimization constraints and objectives returned by the optimizer in case of a vehicle design optimization.

(3) The Central System Intelligence (CSI) manages the whole execution of the program, defining the function calls sequence, the information flows among the different subsystems and between the optimizer and the analysis module, etc.

(4) The First Guess Layer (FGL) contains simple analytical relations and variable default bounds used to obtain reference values and lower/upper boundaries for the optimization variables, if not already specified by the user.

(5) The Failure Detection, Isolation and Recovery (FDIR) system actually does not constitute a separate module of the software, but is spread over all the main routines, with the purpose of identifying errors in the execution of the disciplinary and system analysis.

(6) The SubSystem Module (SSM) implements the different disciplinary analysis. Each SSM is called by the CSI, takes data from the CSD and its outputs upon execution are returned to the CSD. Several SSMs exploit external tools for parts of the disciplinary analysis, in

particular CEA for Propulsion, Missile DATCOM for Aerodynamics and Silhouette/3-View for Geometry. All these external tools are called as binary executables directly within the corresponding routines of the SSM, exploiting input/output files as interfaces.

(7) The Optimization Layer (OL) contains several optimization algorithms. For single-objective optimization problems: a global stochastic algorithm (PSO-1D), a gradient-based local algorithm (SQP solver WORHP) and a combined branch and bound and NLP solver BBWORHP. For multiple objective optimization problems: four global stochastic multi-objective algorithms (DG-MOPSO, NSGA-II MOACOr and their hybridization HybridGO), and the derivative free deterministic MADS algorithm. The call to the algorithms is again managed by the CSI, which defines the structure of the optimization problem starting from the user inputs stored in the XML file which is available via the CSD.

As already mentioned, it is very important to point out that the software has been designed with particular attention to future extensions and maintainability. For this purpose, each SSM communicates directly with the CSD through the CSI, whereas no lateral communications between the different SSMs are allowed as shown in Figure B.2. In this way, the software is made drastically more expandable and maintainable: every time a SSM needs to be modified, substituted or added, only the input/outputs with a single entity (the CSD) have to be coherently defined, without worrying about interfaces with all other SSMs.

The step by step development process, as well as its wide range of applicability and its flexible architecture, inspired the designers to create a single tool capable to tackling different kinds of analysis and optimization problems. SVAGO can therefore be used in 3 different modes:

(1) Frozen Design (FD): this mode only involves the trajectory models of SVAGO, no launch vehicle analysis and design capability is used. The user has to provide all design data required for integrating a trajectory: architecture, dry and wet masses, propulsion performance indexes, and aerodynamic coefficients. Two alternative trajectory analysis can be selected in this mode:
   a) Trajectory Simulation (TS), involving the simple integration of the EoM from take-off to propellant's depletion. The guidance profiles (pitch, yaw and thrust throttle versus time) have to be provided by the user through input files.
   b) Trajectory Optimization (TO), optimization for maximum payload of the launcher's trajectory to reach a given target orbit.

   The first guess can be automatically computed with standard guidance laws or can again be defined by the user through input files.

(2) Multidisciplinary Design Analysis (MDA): this mode allows the user to analyze the design of a launch vehicle, sequentially running all disciplinary codes. The user has to provide all data required for

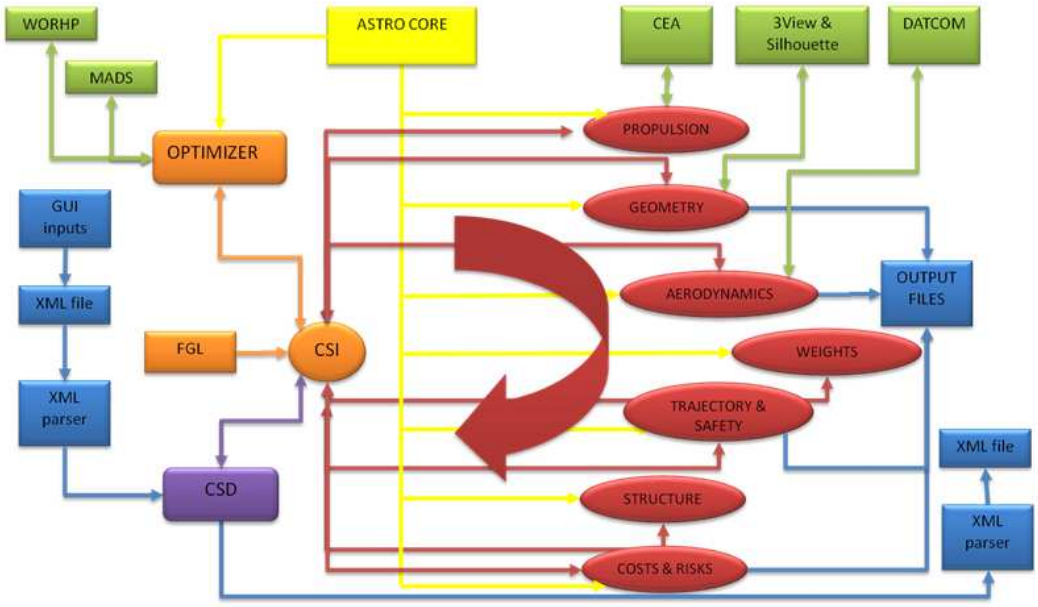


FIGURE B.2. SVAGO software architecture. In red is marked the MDA, in blu the input and output layer, in green the external tool embedded in the software and in yellow the core mathematical routines.

running the disciplinary models, at system and subsystem level. As for the FD mode, the trajectory module can be executed in the two alternative modes of TS and TO. Whereas the latter is identical, TS mode differs with respect to the FD case in the guidance profiles definition: pitch and yaw are obtained using the standard guidance laws, whose parameters can be modified by the user to manually adjust the trajectory.

(3) Multidisciplinary Design Optimization (MDO): this mode allows exploiting the full MDO capabilities of SVAGO. An optimization algorithm recursively calls the MDA model, trying to optimize the architecture and design parameters of the launch vehicle, concurrently with its trajectory. The overall optimization architecture can be single or bi-level. In the single level BBO all design and trajectory optimization variables are treated on the same level by the same optimization algorithm and the trajectory simulation option of the MDA mode is used at each design loop. In the bilevel NOL, a trajectory optimization loop is nested inside each design loop, using the trajectory optimization option of the MDA mode above. In addition to the MDO architecture, the optimization approach can also be chosen by the user among the following:
   a) Global Optimization (GO): single or multi-objective global optimization with one of the available algorithms;

b) Local Refinement (LR): single-objective local optimization with WORHP, starting from a previous globally obtained solution given via an input file, the value of the discrete variables are frozen to the selections obtained in the previous global run;
c) Global Optimization with Local Refinement (GO+LR): combination of the two optimization approaches above in the same run.

Bibliography

[1] AIAA Technical Committee on Multidisciplinary Design Optimization. White Paper on Industrial Experience with MDO. American Institute of Aeronautics and Astronautics. 1998.

[2] Giesing J.P., Barthelemy J. M., A Summary of Industry MDO Applications and Needs. Symposium on Multidisciplinary Analysis and Optimization. 1998.

[3] AIAA Technical Committee on Multidisciplinary Design Optimization. White paper on current state of art. American Institute of Aeronautics and Astronautics. 1991.

[4] Sobieszczanski-Sobieski J., Multidisciplinary Optimization for Engineering Systems: Achievements and Potential. NASA TM 101566, March 1989.

[5] https://info.aiaa.org/tac/adsg/MDOTC, July 2012.

[6] Blair J.C., Ryan R.S., Schutzenhofer L.A. and Humphries W.R., Launch Vehicle Design Process: Characterization, Technical Integration, and Lessons Learned, NASA/TP 2001 210992, May 2001.

[7] Sobieszczanski-Sobieski J., Haftka R.T., Multidisciplinary aerospace design optimization: survey of recent development, Structural Optimization, Vol.14, pp. 1-23, Springer-Verlag 1997.

[8] Humphries W.R., Holland W. and Bishop R., Information Flow in the Launch Vehicle Design/Analysis Process, NASA/TM1999209877, December 1999.

[9] Pilchen, G. M., Breteau J., Caruana J., Kauffmann J., Ramusat G., Sirbi A., and Tumino G., Future Launchers Preparatory Programme (FLPP) - Preparing For The Future Through Technology Maturation And Integrated Demonstrators Status And Perspectives, 59th International Astronautical Congress, IAC-08-D 2.5.2, Glasgow, UK, October 2008.

[10] Advanced Re-Entry Vehicle (ARV), ESA Document ESA-HSO-COU-026, Rev. 2.0.

[11] Cramer E.J., Frank P.D., Shubin G.R., Dennis J.E. Jr. and Lewis R.M., On Alternative Problem Formulations for Multidisciplinary Design Optimization, AIAA Paper 92-4752.

[12] Balling L.J. and Sobieszczanski-Sobieski J., Optimization of Coupled Systems: A Critical Overview of Approaches, AIAA Journal, Vol. 34, No. 1, pp. 6-17, January-February 1996.

[13] Rogers I.L., A Knowledge-Based Tool for Multilevel Decomposition of a Complex Design Problem, NASA TR2903, 1989.

[14] Braun R.D., Powell R.W., Lepsch R.A., Stanley D.O and Kro I.M., Comparison of Two Multidisciplinary Optimization Strategies for Launch-Vehicle Design, Journal of Spacecraft and Rockets, Vol. 32, No. 3, May-June 1995.

[15] Sobieszczanski-Sobieski J., A Linear Decomposition Method for Large Optimization Problems, NASA TM 83248, February 1982.

[16] Sobieszczanski-Sobieski J. and Barthelemy J.F., Improving Engineering System Design by Formal Decomposition, Sensitivity analysis, and Optimization, Proceeding of the International Conference of Engineering Design, Hamburg, West Germany, August 1985.

[17] Wrenn G.A. and Dovi A.R., Multilevel Decomposition Approach to the Preliminary Sizing of a Transport Aircraft Wing, AIAA Journal of Aircraft, Vo1.25. No. 7, pp. 632-638, July 1988.
[18] Bindolino G., Ghiringhelli G.L. and Ricci S., A Multi-Level Procedure For The Preliminary Sizing Of Aircraft Wing-Box Structures, XVIII National AIDAA Congress, September 2005.
[19] Sobieszczanski-Sobieski J., Optimization by Decomposition: a Step from Hierarchic to Non-Hierarchic Systems, NASA report N89-25149, 1989.
[20] Bloebaum C.L., Hajela P. and Sobiesezczanski-Sobieski, J., Non-Hierarchic System Decomposition in Structural Optimization, Engineering Optimization, Vol. 19, pp. 171-186, 1992.
[21] Kreisselmeier G. and Steinhauser R., Systematic Control Design by Optimizing a Vector Performance Index, International Federation of Active Controls Symposium on Computer-Aided Design of Control Systems, Zurich, Switzerland, 1979.
[22] Renaud J.E. and Gabriele G.A., Improved Coordination in Non-Hierarchic System Optimization, AIAA Paper 92-2497-CP, 1992.
[23] Stelmack M.A. and Batill S.M., Concurrent Subspace Optimization of Mixed Continuous/Discrete Systems, AIAA Paper 1997-1229, 1997.
[24] Sobieszczanski-Sobieski J., Agte J.S. and Sandusky R.R.Jr., Bi-Level Integrated System Synthesis (BLISS), NASA TM-1998-208715, August 1998.
[25] Sobieszczanski-Sobieski J., Emiley M.S., Agte J.S. and Sandusky R.R.Jr., Advancement of Bi-Level Integrated System Synthesis (BLISS), NASA TM-2000-210305, December 2000.
[26] De Baets P., Mavris D. and Sobieszczanski-Sobieski J., Aeroelastic Design by Combining Conventional Practice with Bi-Level Integrated System Synthesis (BLISS), AIAA Paper 2004-4431, 2004.
[27] Sobieszczanski-Sobieski J., Altus T.D., Phillips M. and Sandusky R.R.Jr., Bi-Level Integrated System Synthesis (BLISS) for Concurrent and Distributed Processing, AIAA Paper 2002-5409, 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Atlanta, Georgia, September 2002.
[28] Altus T.D., A Response Surface Methodology for Integrated System Synthesis (BLISS), NASA/CR-2002-211652, May 2002.
[29] Simpson T.W., Peplinski J., Koch P.N. and Allen J.K., Metamodels for Computer-Based Engineering Design: Survey and Recommendations, Engineering Computing, Vol. 17, Issue 2, pp. 129-150, 1995.
[30] Kroo I., Altus S., Braun R. Gage P. and Sobieszczanski-Sobieski J., Multidisciplinary Optimization Methods for Aircraft Preliminary Design, AIAA Paper 94-4325-CP, 1994.
[31] Braun R. and Kroo I., Development and Application of the Collaborative Optimization Architecture in a Multidisciplinary Design Environment, Proceedings of the ICASE/NASA Langley Workshop on Multidisciplinary Design Optimization, Hampton, Virginia, March 1995.
[32] Braun R., Moore A.A. and Kroo I., Use of the Collaborative Optimization Architecture for Launch Vehicle Design, 6th Symposium on Multidisciplinary Design and Optimization, Bellevue, WA, September 1996.
[33] Braun R., Collaborative Optimization: an architecture for large-scale distributed design, PhD dissertation, Stanford University, Department of Aeronautics and Astronautics, May 1996.
[34] Braun R., Gage P., Kroo I. and Sobieszczanski-Sobieski J., Implementation and Performance Issues in Collaborative Optimization, AIAA Paper 96-4017, 1996.
[35] Alexandrov M.N. and Kodiyalam S., Initial results of an MDO method evaluation study, AIAA Paper 1998-4884, 1998.

[36] Alexandrov M.N. and Lewis R.M., Comparative Properties Of Collaborative Optimization And Other Approaches To MDO, Proceedings of the First ASMO UK /ISSMO Conference on Engineering Design Optimization, July 1999.
[37] Alexandrov M.N. and Lewis R.M., Analytical and Computational Aspects of Collaborative Optimization, NASA TM 2000-210104, April 2000.
[38] Kroo I. and Manning V., Collaborative Optimization: Status and Directions, AIAA Paper 2000-4721, 2000.
[39] Sobieski I.P., Manning V.M. and Kroo I.M., Response Surface Estimation and Refinement in Collaborative Optimization, AIAA Paper 1998-4753, 1998.
[40] Kroo I., Distributed Multidisciplinary Design and Collaborative Optimization, VKI lecture series on Optimization Methods and Tools for Multicriteria/Multidisciplinary Design, November 2004.
[41] De Miguel A.V. and Murray W., An Analysis of Collaborative Optimization Methods, 8th Symposium on Multidisciplinary Design and Optimization, Long Beach, CA, September 2000.
[42] De Miguel A.V., Two Decomposition Algorithms for Nonconvex Optimization Problems with Global Variables, Ph.D. thesis, Stanford University, 2001.
[43] Roth B. and Kroo I., Enhanced Collaborative Optimization: Application to an Analytic Test Problem and Aircraft Design, 12th Symposium on Multidisciplinary Design and Optimization, Victoria, British Columbia, September 2008.
[44] Deb K., Evolutionary Algorithms for Multi-Criterion Optimization in Engineering Design, Proceedings of Evolutionary Algorithms in Engineering and Computer Science (EUROGEN 99), 1999.
[45] Huang C.H. and Bloebaum C.L., Multi-Objective Pareto Concurrent Subspace Optimization for Multidisciplinary Design, 42nd AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, Jan. 2004.
[46] Rabeau S., Dépincé P., Bennis F. and Janiaut R., Comparison of Global and Local Treatment for Coupling Variables into Multidisciplinary Problems, 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Portsmouth, Virginia, September 2006.
[47] Padula S.L., Alexandrov N. and Green L.L., MDO Test Suite at NASA Langley Research Center, AIAA Paper 96-4028.
[48] Balling L.J. and Wilkinson C.A., Execution of Multidisciplinary Design Optimization Approaches on Common Test Problems, 6th Symposium on Multidisciplinary Design and Optimization, Bellevue, WA, September 1996.
[49] Tedford N.P. and Martins J.R.R.A, Comparison of MDO Architectures within a Universal Framework, AIAA Paper 2006-1617, 47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conferecne, Newport, Rhode Island, May 2006.
[50] Tedford N.P. and Martins J.R.R.A, On the Common Structure of MDO Problems: a Comparison of Architectures, 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Portsmouth, Virginia, September 2006.
[51] Yi S.I., Shin J.K and Park G.J., Comparison of MDO methods with mathematical examples, Structural and Multidisciplinary Optimization, Vol. 35, pp. 391-402, 2008.
[52] Castellini F., Multidisciplinary Design Optimization for Expendable Launch Vehicles, PhD dissertation, Politecnico of Milano, Department of Aeronautics and Astronautics, March 2012.
[53] Pintr J.D., Global Optimization in action, Kluwer Academic Publisher, Springer, 1996.
[54] Horst R., Pardalos P.M., Romeijn H.E., Handbook on Global Optimization: Nonconvex Optimization and Its Applications, Kluwer Academic Publisher, Springer, 1995.
[55] Floudas C.A., Deterministic Global Optimization, Kluwer Academic Publisher, Springer, 2000.

[56] Knowles J., Corne D., Memetic Algorithms for Multiobjective Optimization: Issues, Methods and Prospects, IEEE Press, pp. 325-332, 2000.
[57] Lawler E. L., Wood D. E., Branch-And-Bound Methods: A Survey, Operations Research, Vol.14, No.4, pp. 699-719, 1966.
[58] Falk J.E. and Soland R.M., An algorithm for separable nonconvex programming problems, Management Science, Vol. 15, pp. 550-569, 1969.
[59] Rajasekaran S., On simulated annealing and nested annealing, Journal of Global Optimization, Vol. 16, pp. 4356, 2000.
[60] Locatelli M., Simulated annealing algorithms for continuous global optimization, Journal of Optimization Theory and Applications, Vol. 104, pp. 121-133, 2000.
[61] Chelouah R., Siarry P., Tabu search applied to global optimization, European Journal of Operational Research, Vol. 123, pp. 256-270, 2000.
[62] Back T., Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms, Oxford Univ. Press, 1996.
[63] Pardalos P.M., Enumerative Techniques for Solving Some Nonconvex Global Optimization Problems, OR Spektrum, Vol. 10, pp 29-35, 1988.
[64] Diener I., Trajectory Methods in Global Optimization, Handbook of Global Optimization: Nonconvex Optimization and Its Applications, Kluwer, pp. 649-668, 1995.
[65] Forster W., Homotopy Methods, Handbook of Global Optimization: Nonconvex Optimization and Its Applications, Kluwer, pp. 669-750, 1995.
[66] Horst R., Tuy H., Global Optimization: Deterministic Approaches, 3rd ed, Springer-Verlag, Berlin, 1996.
[67] Levy A. V., and Gomez S., The Tunneling Method Applied to Global Optimization, Numerical Optimization, Philadelphia, pp. 213-244, 1985.
[68] Hansen E.R., Walster G.W., Global Optimization using Interval Analysis, CRC Press, 2003.
[69] Solis F. J., Wets R.J.-B., Minimization by Random Search Techniques, Mathematics of Operations Research, Vol. 6, pp. 19-30, 1981.
[70] Kushner H.J., A Versatile Stochastic Model of a Function of Unknown and Time Varying Form, Journal of Mathematical Analysis and Applications, Vol.9, pp. 379-388, 1962.
[71] Mockus J., On Bayesian Methods of Optimization, Toward Global Optimization, Dixon L.C.W.and G.P. Szegö, 1975.
[72] Rinnooy Kan A.H.G., Timmer G.T., Stochastic methods for global optimization, American Journal of Mathematical and Management Sciences, Vol. 4, pp. 7-40, 1984.
[73] Rinnooy Kan A.H.G., Timmer G.T., Stochastic global optimization methods, part I: clustering methods, Mathematical Programming, Vol. 39, pp. 27-56, 1987.
[74] Rinnooy Kan A.H.G., Timmer G.T., Stochastic global optimization methods, part II: multi level methods, Mathematical Programming , Vol. 39, pp. 57-78, 1987.
[75] Laarhoven Peter J. M. , Aarts Emile H. L. Simulated annealing , Springer, 1987.
[76] Kennedy J., Eberhart R., Particle Swarm Optimization, Neural Networks Proceedings., IEEE International Conference, Vol. 4, pp. 1942-1948, 1995.
[77] Goldberg D. E., Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, 1989.
[78] Marco Dorigo M., Sttzle T.,Ant colony optimization, MIT Press, 2004.
[79] Storn R., Price K., Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization, Vol. 11, pp. 341-359, 1997.
[80] Cochocki A., Unbehauen R., Neural Networks for Optimization and Signal Processing, John Wiley & Sons, New York, 1993.
[81] Glover F., Laguna M., Tabu search, Springer, 1997.
[82] Miettinen K., Nonlinear Multiobjective Optimization, Springer, 1999.

[83] Coello Coello C. A., A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques, Knowledge and Information Systems, Vol 1, pp. 269-308, 1998.
[84] Fonseca C. M., Fleming P.J., An Overview of Evolutionary Algorithms in Multiobjective Optimization, Evolutionary Computing, Vol. 3, No. 1, Pages 1-16, 2007.
[85] Brian J., Ritzel J., Wayland E., Ranjithan S., Using genetic algorithms to solve a multiple objective groundwater pollution containment problem, Water Resources Research, Vol 30, No. 5, pp. 1589-1603, 1994.
[86] Ijiri Y., Management Goals and Accounting for Controls, Amsterdam, 1965.
[87] Chen Y. L., Liu C. C., Multiobjective VAR planning using the goal attainment method, IEE Proceedings on Generation Transmission and Distribution, Vol. 141, No. 3, pp. 227-232, 1994.
[88] Fonseca C. M., Fleming P. J., Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization, in Genetic Algorithms Proceedings of the Fifth International Conference, pp. 416-423, San Mateo, CA, 1993.
[89] Srinivas N., Deb K., Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms,Evolutionary Computation, Vol. 2, No. 3, pp. 221-248, 1994.
[90] Horn J., Nafpliotis N., Multiobjective Optimization using the Niched Pareto Genetic Algorithm, IlliGAL Report n.93005, Univeristy of Illinois, 1993.
[91] Zitzler E., Thiele L., Multiobjective evolutionary algorithms: A comparative case study and the strength pareto, IEEE transactions on Evolutionary Computation, 1999.
[92] Zitzler E., Laumanns M., Thiele L., SPEA2: Improving the Strength Pareto Evolutionary Algorithm, TIK-Report 103, 2001.
[93] Deb K., Agrawal S., Pratap A., Meyarivan T., A fast elitist Multi-Objective Genetic Algorithm: NSGA-II, KanGAL Report No. 200001, 2000.
[94] Knowles J.D. , Corne D.W., Approximating the nondominated front using the Pareto Archived Evolution Strategy, Evolutionary Computation, Vol. 8, No. 2, pp. 149-172, 2000.
[95] Knowles J.D., Corne D.W., Oates M.J., The Pareto Envelope-Based Selection Algorithm for Multiobjective Optimization, Lecture Notes in Computer Science, pp. 839-848, 2000.
[96] Corne D.W., Jerram N.R., Knowles J.D., Oates M.J., PESA-II: Region based selection in evolutionary multiobjective optimization, Proceedings of the Genetic and Evolutionary Computation Conference, 2001.
[97] Coello Coello C.A., Toscano Pulido G., A Micro-Genetic Algorithm for Multiobjective optimization, Lecture Notes in Computer Science, pp. 126-140, 2001.
[98] Coello Coello C.A., Toscano Pulido G., The Micro Genetic Algorithm 2: Towards On-Line. Adaptation in Evolutionary Multiobjective Optimization, Lecture Notes in Computer Science, pp. 75, 2003.
[99] Czyzak P., Pareto Simulated Annealing, a meta-heuristic technique for multiple objective combinatorial problems, Journal of Multi-Criteria Decision Analysis, Journal of Multi-Criteria Decision Analysis, Vol. 7, No. 1., pp. 34-47, 1998.
[100] Coello Coello C.A., Toscano Pulido G., Lechuga M.S., Handling multiple objectives with Particle Swarm Optimization, IEEE Transactions on Evolutionary Computation, Vol. 8, pp. 256 - 279, 2004.
[101] Dorigo M., Socha K., Ant Colony Optimization for continuous domains, 2006.
[102] Nocedal J. Wright S.J., Numerical Optimization, Springer, 2006.
[103] Sedighizadeh D., Masehian E., Particle Swarm Optimization Methods, Taxonomy and Applications, International Journal of Computer Theory and Engineering, Vol. 1, No. 5, pp. 486-502, 2009.

[104] Shi Y., Eberhart R. C., Parameter selection in particle swarm optimization, Evolutionary Programming VII, Lecture Notes in Computer Science, Vol. 1447, pp. 591-600, 1998.
[105] Trelea I. C., The particle swarm optimization algorithm: convergence analysis and parameter selection, Journal Information Processing Letters, Vol. 85 Issue 6, pp. 317 - 325,2003.
[106] Land A. H., Doig A. G., An automatic method of solving discrete programming problems, Econometrica, Vol. 28, No. 3, pp. 497-520, 1960.
[107] Lavagna M., Büskens C., Wöbbekind M., Castellini F., Branch and Bound Technique to Efficiently Solve Control and System Design Problems with Mixed-Integer Variables Domains. 61st International Astronautical Congress, Prag, Tschechien, 2010.
[108] Linderoth J. T., Savelsbergh M. W. P., A Computational Study of Search Strategies for Mixed Integer Programming, INFORMS Journal on Computing, Vol. 11, pp. 173-187, 1997.
[109] Achterberg T., Koch T., Martin A., Branching Rules revisited, ZIB Konrad-Zuse-Zentrum fuer Informationstechnik, 2004.
[110] Leyffer S., Integrating SQP and Branch-and-Bound for Mixed Integer Nonlinear Programming, Kluwer Academic Publisher, 2001.
[111] Borchers B., Mitchell J. E., An improved branch and bound algorithm for mixed integer nonlinear programs, R.P.I. Math. Report No. 200, 1991.
[112] Nowak I., Relaxation and Decomposition Methods for Mixed Integer Nonlinear Programming, International Series of Numerical Mathematics, Vol. 152, 2004.
[113] Abhishek K., Leyffer S., Linderoth J. T., Modeling without categorical variables: A mixed-integer nonlinear program for the optimization of thermal insulation systems, Argonne National Laboratory Argonne, IL, 2007.
[114] Wolpert D. H., Macready W. G., No Free Lunch Theorems for Optimization, IEEE Transiction of Evolutionary Computation, Vol. 1, No. 1, pp. 67-82, 1997.
[115] X.H. Shi,Y.C.Liang ,H.P.Lee ,C.Lu ,L.M.Wang, An improved GA and a novel PSO-GA-based hybrid algorithm, Information Processing Letters, 93, pp. 255-261, 2005.
[116] E. Alfassio Grimaldi, F. Grimaccia, M. Mussetta, P. Pirinol, R. E Zich, A New Hybrid Genetical Swarm Algorithm for Electromagnetic Optimization, ICCEA 2004, International Conference on Computational Electromagnetics and Its Applications, 2004. Proceedings, pp. 157 - 160, 2004.
[117] LY. Tseng and SC. Liang, A Hybrid Metaheuristic for the Quadratic Assignment Problem, Computational Optimization and Applications, 34, pp. 85-113, 2005.
[118] Castellini F., Lavagna M., Erb S., Global Optimization Techniques in Space Mission Design, Politecnico di Milano, MsC Thesis, 2008.
[119] Wagner T., Trautmann H., Martì L., A Taxonomy of Online Stopping Criteria for Multi-Objective Evolutionary Algorithms, EMO 2011, LNCS 6576, Springer-Verlag Berlin Heidelberg, pp. 16-30, 2011.
[120] Martì L., Garcia J., Berlanga A., Molina J.M., An Approach to Stopping Criteria for Multiobjective Optimization evolutionary algorithms, Proceedings of the INternational Congress on Evolutionary Computation, pp. 123-1270, Piscataway, 2009.
[121] Goel T., Stander N., A Non Dominance Based Online Stopping Criterion for Multiojective Evolutionary Algorithm, International journal for Numerical Methods in Engineering, 2010.
[122] http://openmp.org, July 2012.
[123] Audet C., Dennis J. E., Mesh Adaptive Direct Search Algorithms for Constrained Optimization, SIAM Journal on Optimization, Vol. 17, No. 1, pp. 188-217, 2006.
[124] Audet C., Dennis J. E., Analysis of Generalized Pattern Searches, SIAM Journal on Optimization, Vol. 13, pp. 889-903, 2000.

[125] Audet C., Dennis J. E., A Progressive Barrier for Derivative-Free Nonlinear Programming, SIAM Journal on Optimization, Vol. 20, pp. 445-472, 2009.
[126] Audet C., B´echard V., Le Digabel S., Nonsmooth optimization through mesh adaptive direct search and variable neighborhood search, Journal of Global Optimization, Vol. 41, pp. 299-318, 2008.
[127] Audet C., Savard G., Zghal W., Multiobjective Optimization Through a Series of Single-Objective Formulations, SIAM Journal on Optimization, Vo. 19, pp. 188-210, 2008.
[128] Audet C., Dennis J. E., Le Digabel S., Globalization strategies for Mesh Adaptive Direct Search, Computational Optimization and Applications, Vol. 46, pp. 193-215, 2010.
[129] Allison J., Roth B., Kokkolaras M., Kroo I., Papalambros P. Y., Aircraft Family Design Using Decomposition-based Methods, 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 2006.
[130] Audet C., Dennis J. E., Le Digabel S., Parallel Space Decomposition of the Mesh Adaptive Direct Search Algorithm, SIAM Journal on Optimization, Vol. 19, pp. 1150-1170, 2008.
[131] www.worhp.de, July 2012.
[132] Han S. P., A globally convergent method for nonlinear programming, Journal of Optimization Theory and Applications, Vol. 22, pp. 297-309, 1977.
[133] Wilson R.B., A simplicial method for convex programming, PhD thesis, Harvard University, 1963.
[134] Gertz E. M., Wright S. J., Object oriented software for quadratic programming, Transactions on Mathematical Software, Vol. 29, No. 1, pp. 58-81, 2003 .
[135] Ulbrich M., Ulbrich S., Vicente L. N., A globally convergent primal-dual interior-point filter method for nonlinear programming, Mathematical Programming, Vol. 100, pp. 379-410, 2004.
[136] Boltyanskii V. G., Gamkrelidze R.V., Pontryagin L.S., Towards a theory of optimal processes, Reports Acad. Sci. USSR, Vol. 110, No. 1, 1956.
[137] Bryson A.E. Jr., Ho Y. C., Applied Optimal Control, Revised Printing, Hemisphere Publishing Corporation, 1975.
[138] Neustadt L., Optimization: A Theory of Necessary Conditions, Princeton University Press, Princeton, NJ, 1976.
[139] Maurer H., On the Minimum Principle for Optimal Control Problems with State Constraints, Vol. 41, Schriftenreihe des Rechenzentrums der Universität Münster, Münster, 1979.
[140] Hartl R., Sethi S., and Vickson R., A survey of the maximum principles for optimal control problems with state constraints, SIAM, Vol. 37, No. 2, pp. 181-218, 1995.
[141] Betts J.T., Practical Methods for Optimal Control Using Nonlinear Programming, SIAM, 2001.
[142] Chapelle A., Cerf M., Cavaillès A., Reynaud S., Fast and Accurate Launcher Trajectory Integration Using Semianalytical Methods, Proceedings of the 4th ICATT, Madrid, 2010.
[143] Augros P., Perrot L., Fast and accurate integration of launcher trajectory applied to trajectory optimization, AIAA 1998.
[144] Schittkowski K., A Collection of 100 Test Problems for Nonlinear Mixed-Integer Programming (User's Guide), Report, Department of Computer Science, University of Bayreuth, 2010.
[145] Bussieck M.R., Drud A.S., Meeraus A., MINLPLib - A collection of test models for mixed-integer nonlinear programming, GAMS Development Corp., Washington D.C., USA, 2007.

[146] Zitzler E., Deb K., Thiele L., Comparison of Multiobjective Evolutionary Algorithms: Empirical Results, Evolutionary Computation, Vol. 8, No. 2, pp. 173-195, 2000.
[147] Deb K., Mathur A. P., Meyarivan T., Constrained Test Problems for Multi-Objective Evolutionary Optimization, Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization,
[148] Deb K., Multi-objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems, Evolutionary Computation, Vol. 7, No. 3, pp. 205-230, 1999.
[149] Akima H., A New Method of Interpolation and Smooth Curve Fitting Based on Local Procedures, Journal of the Association for Computing Machinery, Vol. 17, No. 4, pp. 589-602, 1970.
[150] Akima H., A Method of Bivariate Interpolation and Smooth Surface Fitting Based on Local Procedures, Numerical Mathematics, Vol. 17, No. 1, 1974.
[151] Exler O., Lehmann T., Schittkowski K., A Comparative Study of SQP-Type Algorithms for Nonlinear and Nonconvex Mixed-Integer Optimization, (submitted for publication - 2011).
[152] Marino J., Chemaly A., Structural Design, Analysis, Optimization, and Cost Modeling using the Adaptive Modeling Language (AML), AIAA-2002-1296, 2002.
[153] Morelle P., Klintworth J., Boss-Quattro and MSC/PATRAN: A New Generation of Openarchitecture Multidisciplinary Optimization Software, Proceedings of the MSC Aerospace Users' Conference, Newport Beach, CA, Nov. 1997.
[154] Kroo I., Manning V., Collaborative Optimization: Status and Directions, Proceedings of the 8th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Long Beach, CA, Sept. 2000.
[155] Adams, B.M., Bohnhoff, W.J., Dalbey, K.R., Eddy, J.P., Eldred, M.S., Gay, D.M., Haskell, K., Hough, P.D., and Swiler, L.P., DAKOTA, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 5.0 User's Manual, Sandia Technical Report SAND2010-2183, December 2010 (Version 5.1).
[156] Design Exploration, ANSYS Inc., Release 12.0, Canonsburg PA., April 2009.
[157] Szedula J.A., FASTPASS - A tool for launch vehicle synthesis, Proceedings of the 6th AIAA/NASA/ISSMO, Symposium on Multidisciplinary Analysis and Optimization, Bellevue, WA, Sept. 1996.
[158] Weston R.P., Townsend J.C., Eidson T.M., Gates R.L., A Distributed Computing Environment for Multidisciplinary Design, Proceedings of the 5th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Panama City Beach, FL, Sept. 1994.
[159] User Manual HEEDS MDO, Red Cedar Technology, Computer-Aided Design, Vol. 7, 2011.
[160] HyperStudy, HyperWorks Altair Engineering Inc., Release 11.0, Troy, MI, 1999.
[161] Hale M.A., Craig J.I., Mistree F., Schrage D.P., DREAMS and IMAGE: A Model and Computer Implementation for Concurrent, Life-Cycle Design of Complex Systems, Concurrent Engineering: Research and Applications, Vol. 4, No. 2, pp. 171-186, 1996.
[162] Egorov-Yegorov I.N., Kretinin G.V., Leshchenko I.A., Kuptcov S.V., Multi-Objective Robust Optimization using IOSO Technology, Proceedings of the International Congress on Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems, Barcelona, 2003.
[163] iSIGHT, Dassault Systèmes Simulia, Release 5.6, Providence, RI, Nov. 2011.
[164] Nielen Stander et al., LS-OPT User's Manuals 4.2, Livemore Software Technology Corporation, Livermore CA, May 2011.
[165] http://www.cyberdynesoft.it, July 2012.
[166] Kingsley G., Siegel J.M., Harrand V.J., Lawrence C., Luker J.J., Development of a Multi-Disciplinary Computing Environment (MDICE), AIAA-98-4738, 1998.

[167] Bigley M., Nelson C., Ryan P., Mason W.H., Tutorials and Examples of Software Integration Techniques for Aircraft Design using ModelCenter, Virginia Center for Innovative Technology and Phoenix Integration Inc. Report MAD 99-06-02 , June 1999.

[168] modeFRONTIER, ESTECO, Version 4.4.1, http://www.esteco.it/, Trieste, Italy, 2008.

[169] Nexus: User Manuals − Keywords, iChrome Ltd, Bristol, 2011.

[170] OptiY Software and Documentation Version 4.0, OptiY e.K, 2010, www.optiy.de.

[171] Optimus 10.4, Noesis Solutions, Leuven, 2011, http://www.noesissolutions.com.

[172] I. Kroo. PASS, Program for Aircraft Synthesis Studies. Software Package, Desktop Aeronautics, Palo Alto, CA, 2005.

[173] Chen, S. Y., SmartDO Version 3.1 Manuals, FEA-Opt Technology, 2011, http://www.fea-optimization.com.

[174] Balabanov V.O., Charpentier C., Ghosh D., Quinn G., Vanderplaats G.N., Venter G., VisualDOC: A Software System for General-Purpose Integration and Design Optimization, Proceedings of the 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Atlanta, Georgia, Sept 2002.